Implementação de um processador de 16 bits (com modificações) baseado na arquitetura MIPS, feito no ambiente virtual LOGISIM. Trabalho desenvolvido para a disciplina de Arquitetura de Computadores sob orientação do Professor Me. Alberto da Silva.

Equipe : Luiz Felipe Duarte Fiuza Pereira
Pedro Henrique Soares Medeiros
João Kevin Gomes Rodrigues
Emerson Eustáquio Santos Rodrigues Versiani

Para o processador foi feita a seguinte divisão de bits:

Tipo R:

3 bits	3 bits	3 bits	3 bits	1 bits	3 bits
OP	RS	RT	RD	SH	FUNC

Tipo I:

3 bits	3 bits	3 bits	7 bits
OP	RS	RT	IMM

Tipo J:

3 bits	13 bits
OP	ADDR

Componentes principais: ULA, Banco de Registradores, Unidade de controle, Memória ROM (Memória de programa) e Memória RAM (Memória de dados).

Especificações:

Memória de programa: 16 bits de dados, 4 bits de endereço (2^4 = 16 instruções).
Memória de dados: 8 bits de dados, 7 bits de endereço(128 células de endereço).

IMM : 7 bits sendo 0 até 127 (00H até 7FH)

SHIFT : 1 bit para indicar se haverá deslocamento , 3 bits para representar o valor do deslocamento (deslocamento de 0 a 7).

JUMP pode ser executado nas linha 0 a 15.

EQUAL pode ser executado nas linha 0 a 15.

Operações realizadas pelo processador:

OP	Tipo	FUNÇÕES	EXEMPLOS DE USO
000	R	RD = RS + RT	add $t0, $s1, $s2;
001	R	RD = RT << SH	shift $t0, $d0, 5;
010	R	RD = RS < RT ? 1 : 0	less $d0, $t0, $s0;
011	I	RT = M[IMM + RS]	load $d0, $zero, 01H;
100	I	M[IMM + RS] = RT	store $t1, $zero, 01H;
101	I	IF(RS == RT) PC=IMM	equal $d0, $d1, 3;
110	I	RT = RS + IMM	addi $t0, $zero, 4;
111	J	JUMP ADDR	jump 6;

Operações com a ULA:

CÓDIGO DAS FUNÇÕES	FUNÇÕES
000	ADD (ADIÇÃO)
001	SUB(SUBTRAÇÃO)
010	MULT(MULTIPLICAÇÃO)
011	DIV(DIVISÃO)
100	AND (PORTA LÓGICA AND)
101	OR (PORTA LÓGICA OR)
110	NOT (INVERSORA)
111	(Operação Vazia)

Foi criado em conjunto um programa na linguagem C++ para converter as instruções para o processador, palavras chaves usadas no tradutor:

Palavra	Função
add	Soma entre registradores
sub	Subtração entre registradores
mult	Multiplicação entre registradores
div	Divisão entre registradores
and	AND entre registradores
or	OR entre registradores
not	INVERSÃO do valor do registrador
shift	Multiplicação com deslocamento de bits
less	Comparação de menor
load	Lê da memória
store	Grava na memória
jump	Salta para determinado endereço
addi	Adição com constantes
equal	Salto condicional com comparação de igualdade

Mapeamento dos Registradores para criação de códigos:

Valor	Tipo	Símbolo
0	Zero	$0 ou $zero
1	RS	$s0
2	RS	$s1
3	RS	$s2
4	RT	$t0
5	RT	$t1
6	RD	$d0
7	RD	$d1

Exemplo de código:

addi $s0, $zero, 10; ***addi $t0, $zero, 0;*** ***addi $d1, $zero, 1;*** ***addi $t0, $t0, 1;*** ***less $d0, $t0, $s0;*** ***equal $d0, $d1, 3;*** ***jump 6;***	*S0 = 0+10* *T0 = 0+0* *D1 = 0+1* *T0 = T0+1* *if(T0<S0) { D0=1 } else { D0=0 }* *if(D0=D1) { jump line 3 }* *enquanto T0 < S0, T0 recebe +1;*

Criar um arquivo "main.txt" para escrever as instruções em assembly, colocá-lo na mesma pasta do "TRADUTOR_ASM_HEX_16BITS.exe".
Executar o tradutor, vai ser gerado 2 arquivos, "instructions_bin_16bits.txt" e "instructions_hex.hex", contendo as intruções
em binário e hexadecimal respectivamente;
Agora é só carregar o arquivo "instructions_hex.hex" na memória de programa do Logisim;