**关于测试单周期CPU的简单方法**

**（特别说明：本表每个同学都必须建立，根据需要添加代码，检查实验时，必须提供！）**

1、测试程序段

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **地址** | **汇编程序** | **指令代码** | | | | | |
| **op（6）** | **rs(5)** | **rt(5)** | **rd(5)/immediate (16)** | **16进制数代码** | |
| **0x00000000** | addiu $1,$0,8 | **001001** | **00000** | **00001** | **0000 0000 0000 1000** | **=** | 24010008 |
| **0x00000004** | ori $2,$0,2 | 001101 | 00000 | 00010 | 0000 0000 0000 0010 | **=** | 34020002 |
| **0x00000008** | add $3,$2,$1 | 000000 | 00010 | 00001 | 00011 00000 100000 | **=** | 00411820 |
| **0x0000000c** | sub $5,$3,$2 | 000000 | 00011 | 00010 | 00101 00000 100010 | **=** | 00622822 |
| **0x00000010** | and $4,$5,$2 | 000000 | 00101 | 00010 | 00100 00000 100100 | **=** | 00a22024 |
| **0x00000014** | or $8,$4,$2 | 000000 | 00100 | 00010 | 01000 00000 100101 | **=** | 00824025 |
| **0x00000018** | sll $8,$8,1 | 000000 | 00000 | 01000 | 01000 00001 000000 | **=** | 00084040 |
| **0x0000001c** | **bne $8,$1,-2 (≠,转18)** | 000101 | 01000 | 00001 | 1111 1111 1111 1110 | **=** | 1501fffe |
| **0x00000020** | slti $6,$2,4 | 001010 | 00010 | 00110 | 0000 0000 0000 0100 | **=** | 28460004 |
| **0x00000024** | slti $7,$6,0 | 001010 | 00110 | 00111 | 0000 0000 0000 0000 | **=** | 28c70000 |
| **0x00000028** | addiu $7,$7,8 | 001001 | 00111 | 00111 | 0000 0000 0000 1000 | **=** | 24e70008 |
| **0x0000002c** | **beq $7,$1,-2 (=,转28)** | 000100 | 00111 | 00001 | 1111 1111 1111 1110 | **=** | 10e1fffe |
| **0x00000030** | sw $2,4($1) | 101011 | 00001 | 00010 | 0000 0000 0000 0100 | **=** | ac220004 |
| **0x00000034** | lw $9,4($1) | 100011 | 00001 | 01001 | 0000 0000 0000 0100 | **=** | 8c290004 |
| **0x00000038** | addiu $10,$0,-2 | 001001 | 00000 | 01010 | 1111 1111 1111 1110 | **=** | 240afffe |
| **0x0000003c** | addiu $10,$10,1 | 001001 | 01010 | 01010 | 0000 0000 0000 0001 | **=** | 254a0001 |
| **0x00000040** | **bltz $10,-2(<0,转3C)** | 000001 | 01010 | 00000 | 1111 1111 1111 1110 | **=** | 0540fffe |
| **0x00000044** | andi $11,$2,2 | 001100 | 00010 | 01011 | 0000 0000 0000 0010 | **=** | 304b0002 |
| **0x00000048** | lui $12, 0xffff | 001111 | 00000 | 01100 | 1111 1111 1111 1111 | **=** | 3c0cffff |
| **0x0000004c** | sra $12, $12, 1 | 000000 | 00000 | 01100 | 01100 00001 000011 | **=** | 000c6043 |
| **0x00000050** | srl $12, $12, 1 | 000000 | 00000 | 01100 | 01100 00001 000010 | **=** | 000c6042 |
| **0x00000054** | sllv $12, $12, $6 | 000000 | 00110 | 01100 | 01100 00000 000100 | **=** | 00cc6004 |
| **0x00000058** | srav $12, $12, $6 | 000000 | 00110 | 01100 | 01100 00000 000111 | **=** | 00cc6007 |
| **0x0000005c** | srlv $12, $12, $6 | 000000 | 00110 | 01100 | 01100 00000 000110 | **=** | 00cc6006 |
| **0x00000060** | xor $15, $15, $15 | 000000 | 01111 | 01111 | 01111 00000 100110 | **=** | 01ef7826 |
| **0x00000064** | xori $13, $13, 1 | 001110 | 01101 | 01101 | 0000 0000 0000 0001 | **=** | 39ad0001 |
| **0x00000068** | **bgtz $13, -2(>0,转64)** | 000111 | 01101 | 00000 | 1111 1111 1111 1110 | **=** | 1da0fffe |
| **0x0000006c** | nor $13, $13, $14 | 000000 | 01101 | 01110 | 01101 00000 100111 | **=** | 01ae6827 |
| **0x00000070** | xor $13, $13, $14 | 000000 | 00101 | 01110 | 01101 00000 100110 | **=** | 01ae6826 |
| **0x00000074** | slt $14, $13, $14 | 000000 | 01101 | 01110 | 01110 00000 101010 | **=** | 01ae702a |
| **0x00000078** | sh $12, 2($1) | 101001 | 00001 | 01100 | 0000 0000 0000 0010 | **=** | a42c0002 |
| **0x0000007c** | lh $12, 2($1) | 100001 | 00001 | 01100 | 0000 0000 0000 0010 | **=** | 842c0002 |
| **0x00000080** | sb $12, 1($1) | 101000 | 00001 | 01100 | 0000 0000 0000 0001 | **=** | a02c0001 |
| **0x00000084** | lb $12, 1($1) | 100000 | 00001 | 01100 | 0000 0000 0000 0001 | **=** | 802c0001 |
| **0x00000088** | **jal 0x00000090** | 000011 |  |  | 00 0000 0000 0000 0000 0010 0100 | **=** | 0c000024 |
| **0x0000008c** | **j 0x000000ac** | 000010 |  |  | 00 0000 0000 0000 0000 0010 1011 | **=** | 0800002b |
| **0x00000090** | multu $13, $9 | 000000 | 01101 | 01001 | 00000 00000 011001 | **=** | 01a90019 |
| **0x00000094** | mfhi $15 | 000000 | 00000 | 00000 | 01111 00000 010000 | **=** | 00007810 |
| **0x00000098** | mflo $16 | 000000 | 00000 | 00000 | 10000 00000 010010 | **=** | 00008012 |
| **0x0000009c** | divu $13, $9 | 000000 | 01101 | 01001 | 00000 00000 011011 | **=** | 01a9001b |
| **0x000000a0** | mfhi $17 | 000000 | 00000 | 00000 | 10001 00000 010000 | **=** | 00008810 |
| **0x000000a4** | mflo $18 | 000000 | 00000 | 00000 | 10010 00000 010010 | **=** | 00009012 |
| **0x000000a8** | **jr $31** | 000000 | 11111 | 00000 | 00000 00000 001000 | **=** | 03e00008 |
| **0x000000ac** | **halt** | 111111 | 00000 | 00000 | 0000000000000000 | **=** | fc000000 |

lui, j, sll, srl, sra, sllv, srlv, srav, addiu, andi, ori, xori, slti, add, sub, and, nor, or, xor, slt, lw, lh, lb, sw, sh, sb, beq, bne, bltz, bgtz, jal, jr, multu, divu, mfhi, mflo

1、将**指令代码初始化到指令存储器**中，直接写入。

1. 初始化PC的值，也就是以上程序段首地址PC=**0x00000000**，以上程序段从**0x00000000**地址开始存放。
2. 运行Xilinx Vivado进行仿真，看波形。
3. **制作coe文件**

**指令存储器设计成8位，也可参照如下设计。**

**ROM**：即指令存储器， CPU的指令文件即由ROM读取，只读。

input.txt程序中input.txt为指令代码文件，存成.coe或.txt文件。

`timescale 1ns / 1ps

module InstructionMemory(

input[31:0] IAddr,

input RW,

output reg[31:0] IDataOut

);

reg[7:0] InstMemory[255:0];

initial begin //此处为绝对地址，注意斜杠方向

$readmemb("C:/Users/wukan/Documents/VIVADO/SingleCPU/input.txt",InstMemory);//文件路径

end

always@(IAddr or RW)begin

if(RW==0)begin

IDataOut= {InstMemory[IAddr],InstMemory[IAddr+1],InstMemory[IAddr+2],InstMemory[IAddr+3]};

end

end

endmodule

**制作coe文件，对rom进行初始化。**

coe文件的格式一般如下：数据之间以逗号隔开，最后一个数据用分号；

第一行、第二行都是标准格式，就第一行等号后面的数字需要改变，这个数字代表该.coe文件里数据采用什么进制，有2进制、10进制和16进制3种。

如果采用系统提供的IP核方式成成Insrom，采用如下方法对**rom进行初始化。**

## 生成和导入Xilinx ROM/RAM的初始化文件.COE

|  |
| --- |
| 这里比较关键，要导入ROM的值。 在LOAD INIT FILE 项上打勾，点击LOAD FILE……出现下图，要你选择一个文件。  这个文件就是你要放在ROM中的数据，文件的后缀名是.COE。你可以自己建一个空的文本文件，然后把后缀名该为.COE即可。  **Rom汇编程序设计、代码**  **1.** 汇编代码main:  addi $s1,$zero,85 // U  sw $s1,0($s0)  addi $s1,$zero,50 // 2  sw $s1,4($s0)  addi $s1,$zero,48 // 0  sw $s1,8($s0)  addi $s1,$zero,49 // 1  sw $s1,12($s0)  addi $s1,$zero,51 // 3  sw $s1,16($s0)  addi $s1,$zero,49 // 1  sw $s1,20($s0)  addi $s1,$zero,51 // 3  sw $s1,24($s0)  addi $s1,$zero,55 // 7  sw $s1,28($s0)  addi $s1,$zero,54 // 6  sw $s1,32($s0)  addi $s1,$zero,56 // 8  sw $s1,36($s0)  add $4,$2,$3 // 将$4 = $2 + $3  lw $4,4($2) //读出$2偏移16个字节的内容的到$4  sw $2,8($2) //将$2写入$2偏移32字节的RAM  sub $2,$4,$3 // $2=$4-$3  or $2,$4,$3 // $2=$4 | $3  and $2,$4,$3 // $2=$4 & $3  slt $2,$4,$3 // $4<$3，$2 = 1  beq $4,$3,exit //$4 == $3，退出  j main  exit:lw $2,0($3)  j main  **2. 机器码**    **3. 制作coe文件**  MEMORY\_INITIALIZATION\_RADIX=16;  MEMORY\_INITIALIZATION\_VECTOR=  20110055  ae110000  20110032  ae110004  20110030  ae110008  20110031  ae11000c  20110033  ae110010  20110031  ae110014  20110033  ae110018  20110037  ae11001c  20110036  ae110020  20110038  ae110024  00432020  8c440004  ac420008  00831022  00831025  00831024  0083102a  10830002  08000000  8c620000  08000000 |

Sim.v

仿真模块。

`timescale 1ns / 1ps

module Sim;

reg CLK; //时钟信号

reg Reset; //置零信号

SingleCPU scpu(CLK,Reset);

initial begin

CLK= 0;

Reset= 0; //刚开始设置pc为0

#50; //等待Reset完成

CLK= !CLK; //下降沿，使PC先清零

#50;

Reset= 1; //清除保持信号

forever #50 begin //产生时钟信号，周期为50s

CLK= !CLK;

end

end

endmodule