Project1 VerilogHDL完成单周期处理器开发

# 设计说明

1. 处理器应实现MIPS-Lite1指令集。
   1. MIPS-Lite1＝{MIPS-Lite，addi，addiu， slt，jal，jr}。
   2. MIPS-Lite指令集：addu，subu，ori，lw，sw，beq，lui，j。
   3. addi应支持溢出，溢出标志写入寄存器$30中第0位。
2. 处理器为单周期设计。

# 设计要求

1. 单周期处理器由datapath(数据通路)和controller(控制器)组成。
   1. 数据通路由如下module组成：PC(程序计数器)、NPC(NextPC计算单元)、GPR (通用寄存器组，也称为寄存器文件、寄存器堆)、ALU(算术逻辑单元)、EXT(扩展单元)、IM(指令存储器)、DM(数据存储器)。
   2. IM：容量为1KB(8bit×1024)。
   3. DM：容量为1KB(8bit×1024)，采用小端序方式存取数据。
2. Figure1为供你参考的数据通路架构图。
   1. 我们不确保Figure1是完全正确的；我们也不确保Figure1能够满足MIPS-Lite1。
   2. 鼓励你从数据通路的功能合理划分的角度自行设计更好的数据通路架构。

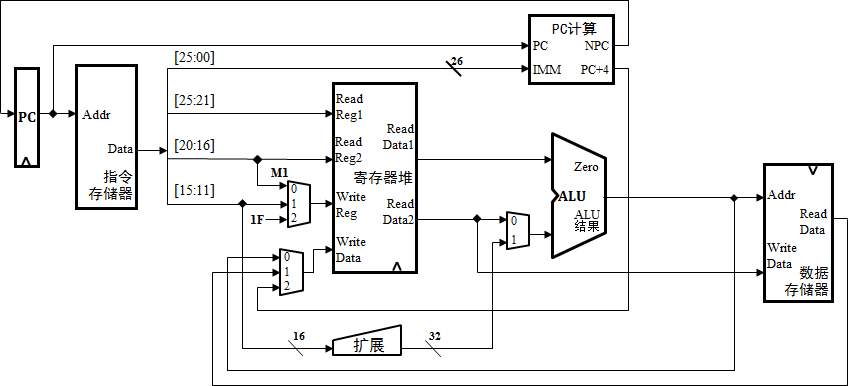


Figure1 数据通路(供参考)

1. 使用code.txt文件存储指令码
   1. 用VerilogHDL建模IM时，必须以读取文件的方式将code.txt中指令加载至IM中。
   2. code.txt的格式如Figure2所示。每条指令占用4行，指令二进制码以文本方式存储。



Figure2 code.txt文件格式

1. 为使得代码更加清晰可读，建议多使用宏定义，并将宏定义组织在合理的头文件中。
2. PC复位后初值为0x0000\_3000，目的是与MARS的Memory Configuration相配合。
   1. 测试程序将通过MARS产生，其配置模式如Figure3所示。



Figure3 MIPS存储配置模式(MARS memory configuration)

1. 下列模块必须严格满足如下的接口定义：
   1. 你必须在VerilogHDL设计中建模这3个模块。
   2. 不允许修改模块名称、端口各信号以及变量的名称/类型/位宽。

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| mips.v | module mips(clk, rst) ;  input clk ; // clock  input rst ; // reset |
| im.v | im\_1k( addr, dout ) ;  input [11:0] addr ; // address bus  output [31:0] dout ; // 32-bit memory output  reg [7:0] im[1023:0] ; |
| dm.v | dm\_1k( addr, din, we, clk, dout ) ;  input [11:0] addr ; // address bus  input [31:0] din ; // 32-bit input data  input we ; // memory write enable  input clk ; // clock  output [31:0] dout ; // 32-bit memory output  reg [7:0] dm[1023:0] ; |

# 测试要求

1. 所有指令都应被测试充分。
2. 构造至少包括40条以上指令的测试程序，并测试通过。
   1. MIPS-Lite1定义的每条指令至少出现1次以上。
   2. 必须有函数，并至少1次函数调用。
3. 函数相关指令(jal和jr)是较为复杂的指令，其正确性不仅涉及到自身的正确性，还与堆栈调整等操作相关。因此为了更充分的测试，你必须在测试程序中组织一个循环，并在循环中多次函数调用，以确保正确实现了这2条指令。

# 其他要求

1. 提交的压缩文件内容：工程中所有.v文件、code.txt、code.asm、项目报告。
2. 时间要求：各班实验指导教师指定。

# 成绩及实验测试要求

1. 实验成绩包括但不限于如下内容：初始设计的正确性、增加新指令后的正确性、实验报告等。
2. 实验测试时，你必须已经完成了处理器设计及开发。
3. 实验测试时，你需要展示你的设计并证明其正确性。
   1. 应简洁的描述你的验证思路，并尽可能予以直观展示。
4. 实验指导教师会临时增加1～2条指令，你需要在规定时间内完成对原有设计的修改，并通过实验指导教师提供的测试程序。
   1. 考查时，教师将用专用testbench和code.txt检测代码执行情况。

# 开发与调试技巧

1. 对于每条指令，请认真阅读《MIPS32® Architecture For Programmers Volume II: The MIPS32® Instruction Set》！
   1. 如果测试时，你无法清楚的解释所要求的指令，测试成绩将减一档！
2. 利用$readmemh系统任务可以给存储器初始化数据。例如可以把code.txt文件中的数据加载至my\_memory模块。

|  |
| --- |
| reg [7:0] my\_memory[1023:0] ;  initial  $readmemh( “code.txt”, my\_memory ) ; |

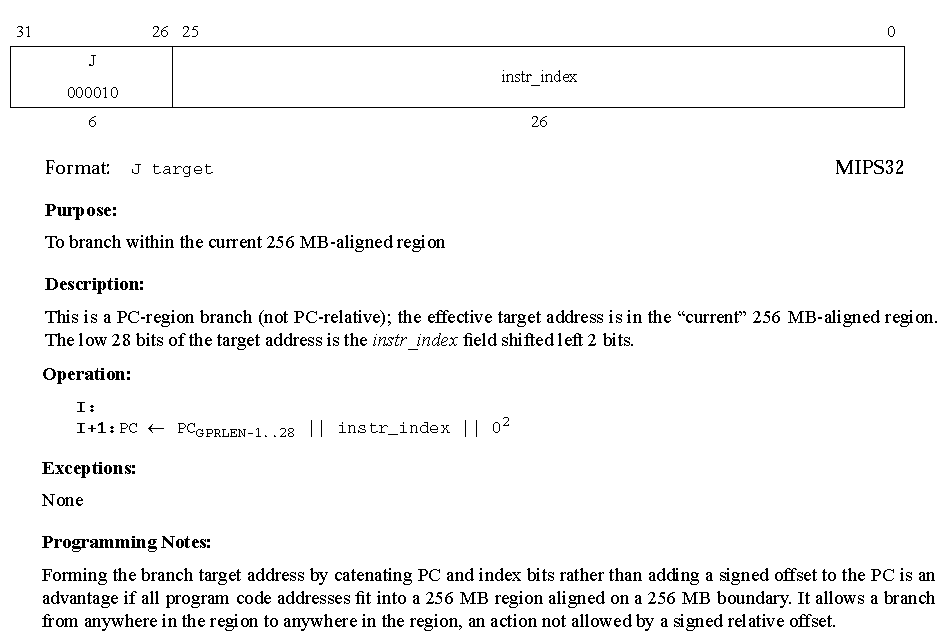
1. 有时我们需要较为集中的在顶层testbench中观察甚至修改下层模块的变量，那么你可以通过使用层次路径名来非常方便的达到这一目的。例如：

|  |
| --- |
| module testbench ;  Chil C1(…) ;  $display(C1.Art) ;  endmodule  module Chil(…) ;  reg Art;  …  endmodule |

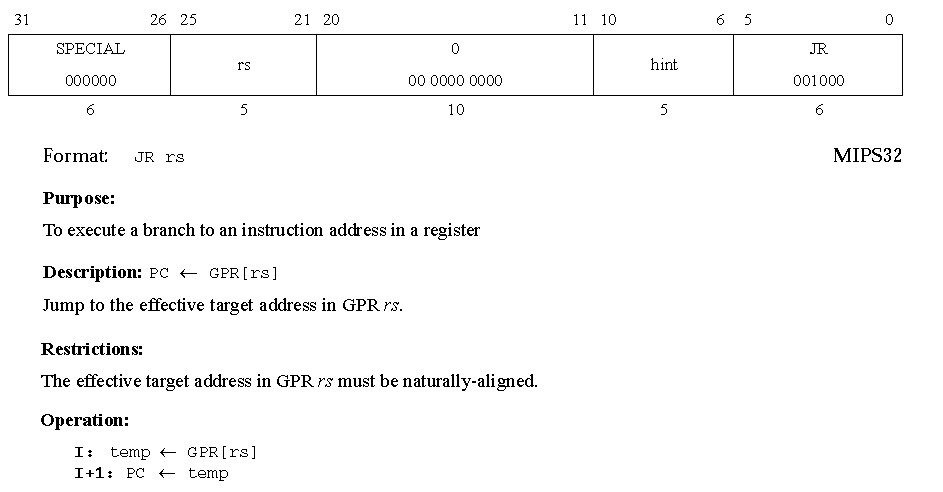
# 补充说明

由于MIPS-Lite1部分指令(主要是JAL、JR)涉及非常复杂的运行模式，故你在阅读《MIPS32® Architecture For Programmers Volume II: The MIPS32® Instruction Set》时可能存在困难。为此我们做了简化处理以便于你理解。简化主要是去除所有与exception、delay slot有关的描述。J、JAL、JR所有简化后的描述陈列如下。

1. J指令



1. JR指令



1. JAL指令

