Project2 VerilogHDL完成流水线处理器开发

# 设计说明

1. 完成以下指令集。
   1. 指令：

add，addu，sub，subu，and，or，nor，xor，slt，sltu

addi，addiu，andi，ori，~~nori~~，xori，slti，sltiu

Lui sll，srl，sra

lw，lh，lhu，lb，lbu，sw，sh，sb

beq，bne

J，jal，jr

新增：

addu，subu，nor，sltu（无符号数比较）

andi，ori，~~nori~~，xori（含or的都是0扩展），slti（带符号数比较），sltiu（无符号数比较）

sll，srl，sra（算数右移）（移位运算中，只有算术右移符号扩展，其他都补0）

（前提：若cpu与mm都采用小端方式存放数据）

lh（都是符号扩展），lhu（半字0扩展，offset符号扩展），lb（都是符号扩展），lbu（字节0扩展，offset符号扩展）

sh，sb（跟sw没区别）

bne

jal，jr

共34条

* 1. 不支持溢出。

1. 处理器为流水线设计，无需支持各类冒险与转发。

# 设计要求

1. 流水线处理器由datapath(数据通路)和controller(控制器)组成。
   1. 数据通路在单周期功能部件的基础上，增加了相应的流水寄存器。
   2. IM：容量为2KB(32bit×512字)。
   3. DM：容量为4KB(32bit×1024字)。
2. Figure1为供你参考的处理器架构图。
   1. 我们不确保Figure1是完全正确的；我们也不确保Figure1能够满足上述指令集。
   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。
   3. 如果你做了比较大的调整，请务必注意不要与要求5矛盾。

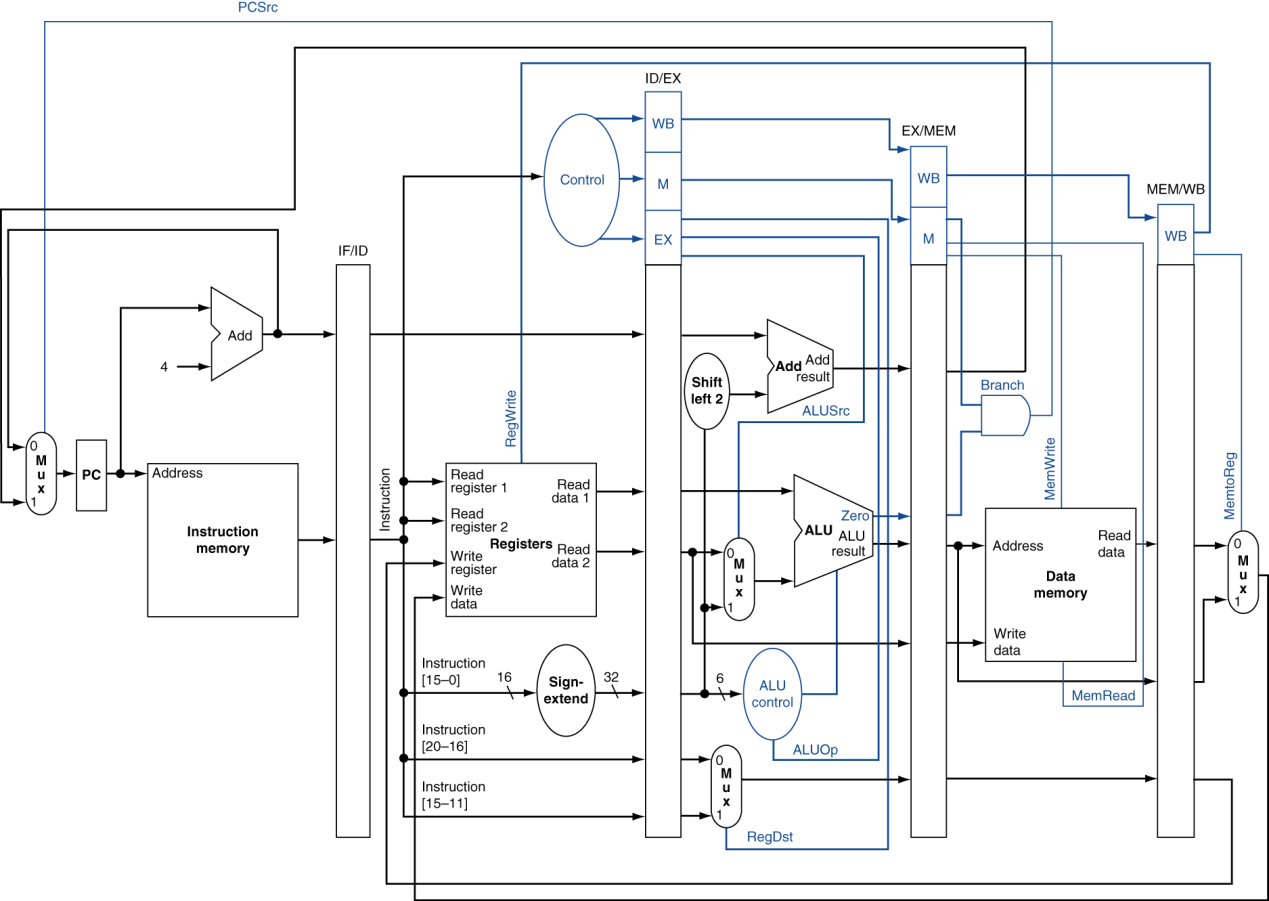


Figure1数据通路(供参考)

1. 整个project必须采用模块化和层次化设计。

* 1. Figure2为参考的目录结构和文件命名。其中红色框的目录名称及文件名称不允许调整(control、datapath、mips.v、code.txt都属于同一层；control目录下包括ctrl.v；datapath目录下包括im.v、dm.v，等等)。

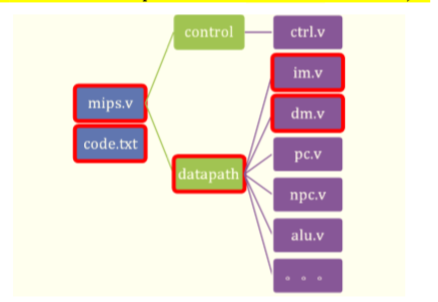


Figure2参考的project目录组织

* 1. 顶层设计文件命名为mips.v。
  2. 建议datapath中的每个module都由一个独立的VerilogHDL文件组成。
  3. 控制器的设计与单周期非常相似，除了要保留对应的控制信号到下一级流水阶段。

1. code.txt中存储的是指令码
   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Figure3所示。每条指令占用1行，指令二进制码以文本方式存储。

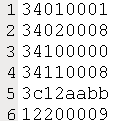


Figure3code.txt文件格式

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. 教师用测试程序将通过MARS产生，其配置模式如Figure4所示。



Figure4MIPS存储配置模式(MARS memory configuration)

# 模块定义【WORD】

1. 仿照下面给出的PC模块定义，给出所有功能部件的模块定义。
2. PC模块定义(参考样例)
3. 基本描述

PC主要功能是完成输出当前指令地址并保存下一条指令地址。复位后，PC指向0x0000\_3000，此处为第一条指令的地址。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| NPC[31:2] | I | 下条指令的地址 |
| clk | I | 时钟信号 |
| Reset | I | 复位信号。  1：复位  0：无效 |
| PC[31:2] | O | 30位指令存储器地址(最低2位省略) |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时，PC被设置为0x0000\_3000。 |
| 2 | 保存NPC并输出 | 在每个clock的上升沿保存NPC，并输出。 |

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\_4k( addr, dout ) ;  input [11:2] addr ; // address bus  output [31:0] dout ; // 32-bit memory output  reg [31:0] im[1023:0] ; |
| dm.v | dm\_4k( addr, din, we, clk, dout ) ;  input [11:2] 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 [31:0] dm[1023:0] ; |

# 测设要求

1. 所有指令都应被测试充分。
2. 构造至少包括10条以上指令的测试程序，并测试通过，每条指令至少出现1次以上。
3. 详细说明你的测试程序原理及测试结果。【WORD】
   1. 应明确说明测试程序的测试期望，即应该得到怎样的运行结果。
   2. 每条汇编指令都应该有注释。

# 问答【WORD】

1. C语言是一种弱类型程序设计语言。C语言中不对计算结果溢出进行处理，这意味着C语言要求程序员必须很清楚计算结果是否会导致溢出。因此，如果仅仅支持C语言，MIPS指令的所有计算指令均可以忽略溢出。

# 其他要求

1. 打包文件：VerilogHDL工程文件、code.txt、code.txt所对应的汇编程序、项目报告。
2. 时间要求：5月17号or5月24号？
3. 本实验要求文档中凡是出现了【WORD】字样，就意味着该条目需要在实验报告中清晰表达。

# 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。
   1. 允许实验报告可以未完成。
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。

# 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. 建议先在MARS中编写测试程序并调试通过。注意memory configuration的具体设置。
   1. 你应该加载code.txt至指令存储器以测试你的处理器设计。假设你的处理器设计是正确的。
   2. 你需要参照Figure4设置MARS，否则该程序将无法运行。
3. 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 [31:0] 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 ;  ChilC1(…) ;  $display(C1.Art) ;  endmodule  moduleChil(…) ;  reg Art;  …  endmodule |