# Project3 VerilogHDL 完成流水线处理器开发

## 一、 设计说明

1. 完成以下指令集。
   a) 如单周期一样的 14 条指令：add，addi，addiu, sub，and，or，ori，xor，lui，slt，lw，sw，beq，J。
   b) 支持溢出及相应的异常检测；
   c) 支持未定义指令异常。
2. 处理器为流水线设计，支持各类冒险与转发

## 二、 设计要求

3. 流水线处理器由 datapath(数据通路)和 controller(控制器)组成。
   a) 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。
   b) IM：容量为 4KB(32bit×512 字)。
   c) DM：容量为 4KB(32bit×1024 字)。
4. Figure1（请注意，这里并没有增加相应的溢出检测信号，建议自行添加）为供你参考的处理器架构图。
   a) 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。
   b) 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。
   c) 如果你做了比较大的调整，请务必注意不要与要求 5 矛盾。

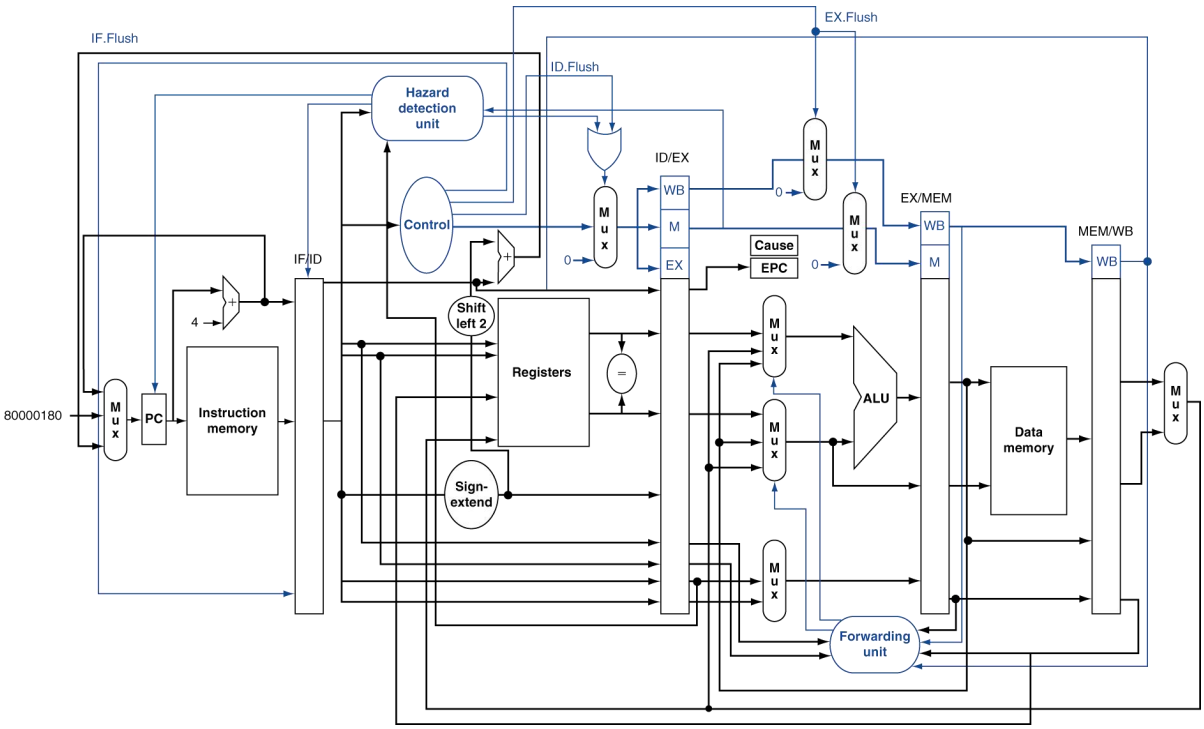


Figure1 数据通路(供参考)

5. 整个 project 必须采用模块化和层次化设计。

a) 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)。

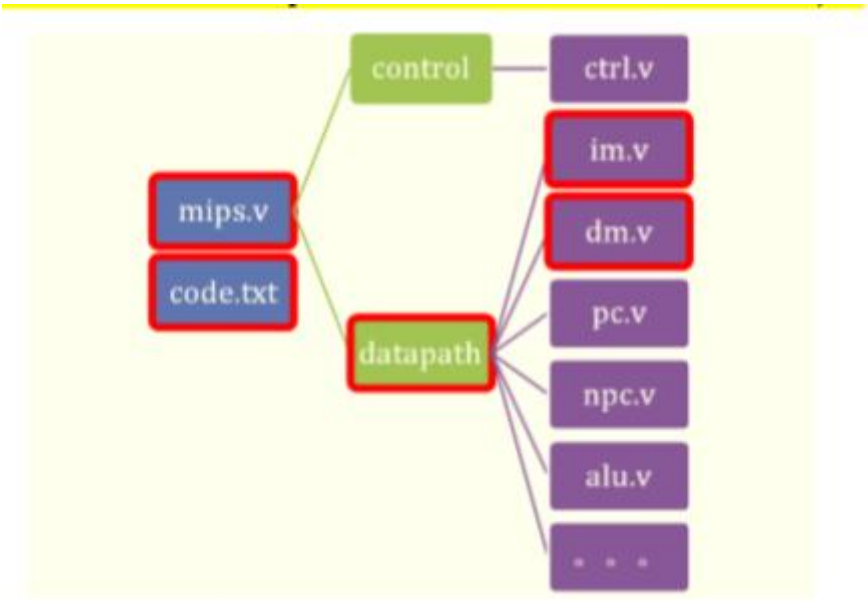


Figure2 参考的 project 目录组织

b) 顶层设计文件命名为 mips.v。

c) 建议 datapath 中的每个 module 都由一个独立的 VerilogHDL 文件组成。

d) 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。

6. code.txt 中存储的是指令码

   a) 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。

   b) 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。

```
1 34010001
2 34020008
3 34100000
4 34110008
5 3c12aabb
6 12200009
```

Figure3code.txt 文件格式

7. 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。

8. PC 复位后初值为 0x0000_3000，目的是与 MARS 的 Memory Configuration 相配合。

   a) 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)

## 三、 模块定义【WORD】

9. 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。

10. PC 模块定义(参考样例)

### (1) 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。

### (2) 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。<br>1：复位<br>0：无效 |
| PC[31:2] | O | 30 位指令存储器地址(最低 2 位省略) |

(3) 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。 |

11. 下列模块必须严格满足如下的接口定义：

a) 你必须在 VerilogHDL 设计中建模这 3 个模块。

b) 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) ;`<br>`input clk ; // clock`<br>`input rst ;// reset` |
| im.v | `im_4k( addr, dout ) ;`<br>`input [11:2] addr ; // address bus`<br>`output [31:0] dout ; // 32-bit memory output`<br><br>`reg [31:0] im[1023:0] ;` |
| dm.v | `dm_4k( addr, din, we, clk, dout ) ;`<br>`input [11:2] addr ; // address bus`<br>`input [31:0] din ; // 32-bit input data`<br>`input we ; // memory write enable`<br>`input clk ; // clock`<br>`output [31:0] dout ; // 32-bit memory output`<br><br>`reg [31:0] dm[1023:0] ;` |

## 四、 测设要求

12. 所有指令都应被测试充分。
13. 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。
14. 详细说明你的测试程序原理及测试结果。【WORD】

a) 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。

b) 每条汇编指令都应该有注释。

## 五、 问答【WORD】

15. 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。

## 六、 其他要求

16. 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。
17. 时间要求：参见在线实验平台时间。
18. 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。

## 七、 成绩及实验测试要求

19. 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。
20. 实验测试时，你必须已经完成了处理器设计及开发。
    a) 允许实验报告可以未完成。
21. 实验测试时，你需要展示你的设计并证明其正确性。
    a) 应简洁的描述你的验证思路，并尽可能予以直观展示。
22. 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。

## 八、 开发与调试技巧

23. 对于每条指令，请认真阅读《MIPS32® Architecture For Programmers Volume II: The MIPS32® Instruction Set》!
    a) 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!
24. 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。
    a) 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。
    b) 你需要参照 Figure4 设置 MARS，否则该程序将无法运行。
25. 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 ) ;
```

26. 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
```