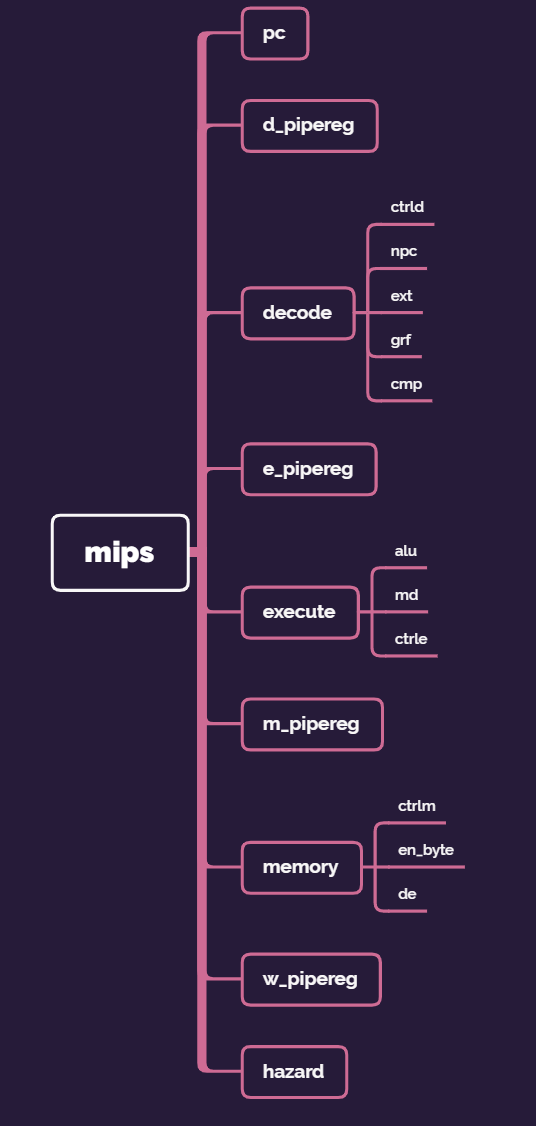
# 计算机组成原理实验报告参考模板

## 一、CPU设计方案综述

### （一）总体设计概述

本CPU为Verilog实现的流水线MIPS - CPU，支持的指令集包含{LB、LBU、LH、LHU、LW、SB、SH、SW、ADD、ADDU、 SUB、 SUBU、 MULT、 MULTU、 DIV、 DIVU、 SLL、 SRL、 SRA、 SLLV、 SRLV、SRAV、AND、OR、XOR、NOR、ADDI、ADDIU、ANDI、ORI、 XORI、LUI、SLT、SLTI、SLTIU、SLTU、BEQ、BNE、BLEZ、BGTZ、 BLTZ、BGEZ、J、JAL、JALR、JR、MFHI、MFLO、MTHI、MTLO}。为了实现这些功能，CPU主要包含了IM、GRF等模块，这些模块按照以下顶层设计逐级展开:



### （二）关键模块定义

#### 1.PC

（1） 端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步置零 |
| 3 | NPC | I | 下一条指令在IM的地址 |
| 4 | en | I | 阻塞信号 |
| 5 | PC | O | 当前指令在IM的地址 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 存储指令地址 | 保存当前执行指令在IM中的地址 |
| 2 | 阻塞 | 阻塞为0时，PC不工作 |

#### 2. D\_pipereg

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号 |
| 3 | en | I | 阻塞信号 |
| 4 | F\_PC | I | F级的PC |
| 5 | F\_instr | I | F级的指令 |
| 6 | D\_PC | O | D级的PC |
| 7 | D\_instr | O | D级的指令 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 存储要流水的值 | 存储要流水的值 |

#### 3. NPC

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | PCSrc | I | 分支信号 |
| 2 | PCj | I | j型跳转信号 |
| 3 | PCjr | I | jr和jalr跳转信号 |
| 4 | D\_PC[31:0] | I | F级PC |
| 5 | F\_PC[31:0] | I | F级PC |
| 6 | Regjr[31:0] | I | jr和jalr跳转的地址 |
| 7 | instr\_index[25:0] | I | D级26位立即数 |
| 8 | NPC[31:0] | O | 下一个PC的值 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 计算下一个PC | 计算下一个PC |

#### 4. EXT

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | imm | I | 16位立即数 |
| 2 | sign | I | 符号扩展选择信号  0：进行无符号扩展  1：进行符号扩展 |
| 3 | SignImm | O | 扩展后的32位数 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 无符号扩展 | 当sign为0时，进行无符号扩展 |
| 2 | 符号扩展 | 当sign为1时，进行有符号扩展 |

#### 5. GRF

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 异步复位信号，  1：清零  0：保持 |
| 3 | PC[31:0] | I | D级PC的值，用于display |
| 4 | A1[4:0] | I | 5位地址输入，从外界接收[25：21]，代表32个寄存器中的一个，并将其中的值输出到RD1 |
| 5 | A2[4:0] | I | 5位地址输入，从外界接收[20：16]，代表32个寄存器中的一个，并将其中的值输出到RD2 |
| 6 | A3[4:0] | I | 5位地址输入，选择32个寄存器中的一个，将WD输入的数据存储到其中 |
| 7 | WD[31:0] | I | 32位写入数据 |
| 8 | RD1[31:0] | O | 输出A1代表的寄存器中的值 |
| 9 | RD2[31:0] | O | 输出A2代表的寄存器中的值 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 异步复位 | Reset为1时，所有寄存器被清零 |
| 2 | 读数据 | 将A1和A2对应的寄存器中的值输出到RD1和RD2 |
| 3 | 写数据 | 如果WE为1，时钟上升沿时将WD中的数据写入到A3对应的寄存器中 |

#### 6. CMP

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | A | I | 操作数A |
| 2 | B | I | 操作数B |
| 3 | D\_equal | O | 相等信号 |
| 4 | D\_equal\_0 | O | 是否等于0 |
| 5 | D\_great\_0 | O | 是否大于0 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 判断A、B是否相等 | 若A=B，则输出1 |
| 2 | 判断A是否等于0 | 若A=0，则输出1 |
| 3 | 判断A是否大于0 | 若A>0，则输出1 |

#### 7. E\_pipereg

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号 |
| 3 | en | I | 阻塞信号 |
| 4 | flush | I | 清洗信号 |
| 5 | D\_PC[31:0] | I | D级PC值 |
| 6 | D\_instr[31:0] | I | D级指令 |
| 7 | D\_signimm[31:0] | I | D级扩展后32位数 |
| 8 | D\_RD1[31:0] | I | D级GRF输出1 |
| 9 | D\_RD2[31:0] | I | D级GRF输出2 |
| 10 | D\_A3[4:0] | I | D级A3地址 |
| 11 | D\_WD[31:0] | I | D级GRF写入值 |
| 12 | E\_PC[31:0] | O | E级PC值 |
| 13 | E\_instr[31:0] | O | E级指令 |
| 14 | E\_signimm[31:0] | O | E级扩展后32位数 |
| 15 | E\_RD1[31:0] | O | E级RD1 |
| 16 | E\_RD2[31:0] | O | E级RD2 |
| 17 | E\_A3[4:0] | O | E级A3地址 |
| 18 | E\_WD[31:0] | O | E级WD |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 存储要流水的值 | 存储要流水的值 |

#### 8. ALU

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | op[2:0] | I | 选择执行哪一个操作  000：加法  001：减法  010：与  011：或  100：左移16位 |
| 2 | inA[31:0] | I | 参与运算的第一个数 |
| 3 | inB[31:0] | I | 参与运算的第二个数 |
| 4 | ALUResult[31:0] | O | 输出inA和inB操作后的结果 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 加法 | 将两个输入加起来输出到ALUResult |
| 2 | 减法 | 将两个输入相减输出到ALUResult |
| 3 | 与运算 | 对两个输入进行与操作输出结果到ALUResult |
| 4 | 或运算 | 对两个输入进行或操作输出结果到ALUResult |
| 5 | 左移16位 | 将SrcB左移16位输出到ALUResult |
| 6 | 或非 | 将两个输入的或非输出到ALUResult |
| 7 | 异或 | 将两个输入的异或输出到ALUResult |
| 8 | 逻辑左移 | 将两个输入的逻辑左移输出到ALUResult |
| 9 | 逻辑右移 | 将两个输入的逻辑右移输出到ALUResult |
| 10 | 算术右移 | 将两个输入的算术右移输出到ALUResult |
| 11 | 小于立即数置1（有符号） | 若A小于B，输出置1 |
| 12 | 小于立即数置1（无符号） | 若A小于B，输出置1 |

#### 9. M\_pipereg

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号 |
| 3 | flush | I | 清洗信号 |
| 4 | E\_PC[31:0] | I | E级PC |
| 5 | E\_WD[31:0] | I | E级WD |
| 6 | E\_A3[4:0] | I | E级A3 |
| 7 | E\_instr[31:0] | I | E级instr |
| 8 | E\_ALUResult[31:0] | I | E级ALUResult |
| 9 | E\_RD2[31:0] | I | E级RD2 |
| 10 | M\_PC[31:0] | O | M级pc |
| 11 | M\_WD[31:0] | O | M级WD |
| 12 | M\_A3[4:0] | O | M级A3 |
| 13 | M\_instr[31:0] | O | M级instr |
| 14 | M\_ALUResult[31:0] | O | M级ALUResult |
| 15 | M\_RD2[31:0] | O | M级RD2 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 存储要流水的值 | 存储要流水的值 |

### 10. en\_byte

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | reset | I | 同步复位信号 |
| 2 | Din | I | 带扩展的数据 |
| 3 | WE | I | 写入信号 |
| 4 | DMaddr | I | DM的读取地址 |
| 5 | Dout | O | 扩展后的数据 |
| 6 | width | I | 是半字还说字节存取 |
| 7 | m\_data\_byteen | O | 要求输出的信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 整字 | 整个完整输出 |
| 2 | 半字 | 按半字规则输出 |
| 3 | 字节 | 按字节读取规则输出 |

### 11. data\_extend

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | DMaddr | I | ALU计算出来的地址 |
| 2 | dataop | I | 数据扩展控制码  000：无扩展  001：无符号字节数据扩展  010：符号字节数据扩展  011：无符号半字数据扩展  1000：符号半字数据扩展 |
| 3 | Din | I | 输入32位数据 |
| 4 | Dout | O | 扩展后的32位数据 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 无扩展 | 无扩展 |
| 2 | 无符号字节数据扩展 | 无符号字节数据扩展 |
| 3 | 符号字节数据扩展 | 符号字节数据扩展 |
| 4 | 无符号半字数据扩展 | 无符号半字数据扩展 |
| 5 | 符号半字数据扩展 | 符号半字数据扩展 |

#### 12. W\_pipereg

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号 |
| 3 | M\_PC[31:0] | I | M级PC |
| 4 | M\_instr[31:0] | I | M级instr |
| 5 | M\_A3[4:0] | I | M级A3 |
| 6 | M\_WD[31:0] | I | M级WD |
| 7 | W\_PC[31:0] | O | W级PC |
| 8 | W\_instr[31:0] | O | W级instr |
| 9 | W\_A3[4:0] | O | W级A3 |
| 10 | W\_WD[31:0] | O | W级WD |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 存储要流水的值 | 存储要流水的值 |

#### 13. Hazard

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号 |
| 3 | D\_A1[4:0] | I | D级读取rs的序号 |
| 4 | D\_A2[4:0] | I | D级读取rt的序号 |
| 5 | D\_RD1[31:0] | I | D级读取rs的值 |
| 6 | D\_RD2[31:0] | I | D级读取rt的值 |
| 7 | D\_rs\_Tuse | I | D级是否在使用rs |
| 8 | D\_rt\_Tuse | I | D级是否在使用rt |
| 9 | E\_A1[4:0] | I | E级读取rs的序号 |
| 10 | E\_A2[4:0] | I | E级读取rt的序号 |
| 11 | E\_RD1[31:0] | I | E级读取rs的值 |
| 12 | E\_RD2[31:0] | I | E级读取rt的值 |
| 13 | E\_A3[4:0] | I | E级写入寄存器的序号 |
| 14 | E\_WD[31:0] | I | E级写入寄存器的值 |
| 15 | E\_rs\_Tuse | I | E级是否使用rs |
| 16 | E\_rt\_Tuse | I | E级是否使用rt |
| 17 | M\_A2[4:0] | I | M级读取rt的序号 |
| 18 | M\_RD2[31:0] | I | M级读取到寄存器的值 |
| 19 | M\_A3[4:0] | I | M级写入寄存器的序号 |
| 20 | M\_WD[31:0] | I | M级写入寄存器的值 |
| 21 | W\_A3[4:0] | I | W级写入寄存器的序号 |
| 22 | W\_WD[31:0] | I | W级写入寄存器的值 |
| 23 | D\_Forward1[31:0] | O | 转发给D级的值1 |
| 24 | D\_Forward2[31:0] | O | 转发给D级的值2 |
| 25 | E\_Forward1[31:0] | O | 转发给E级的值1 |
| 26 | E\_Forward2[31:0] | O | 转发给E级的值2 |
| 27 | M\_Forward2[31:0] | O | 转发给M级的值2 |
| 28 | F\_en | O | PC的使能信号 |
| 29 | D\_pipereg\_en | O | D级流水线寄存器的使能信号 |
| 30 | E\_pipereg\_en | O | E级流水线寄存器的使能信号 |
| 31 | E\_pipereg\_flush | O | E级流水线寄存器的清洗信号 |
| 32 | M\_pipereg\_flush | O | M级流水线寄存器的清洗信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 产生各转发值 | 产生各转发值 |
| 2 | 产生各阻塞信号 | 产生各阻塞信号 |
| 3 | 产生各清洗信号 | 产生各清洗信号 |

#### 14. Control（分布式译码）

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | instr[31:0] | I | 每一级的指令 |
| 2 | D\_equal | I | D级RD1和RD2是否相等 |
| 3 | sign | O | ext是否进行有符号扩展 |
| 4 | PCSrc | O | 是否进行分支 |
| 5 | PCj | O | 是否为J型指令 |
| 6 | PCjr | O | 是否为jr或jalr |
| 7 | D\_WDsel | O | D级中写入GRF值的选择信号 |
| 8 | D\_A3[4:0] | O | D级中写入GRF地址的选择信号 |
| 9 | ALUControl[3:0] | O | ALU选择哪种计算方式 |
| 10 | ALUASel | O | ALU里面inA选择哪个作为输入 |
| 11 | ALUBSel | O | ALU里面inB选择哪个作为输入 |
| 12 | E\_WDSel[1:0] | O | E级中写入GRF值的选择信号 |
| 13 | MemWrite | O | E级中写入GRF地址的选择信号 |
| 14 | width[1:0] | O | 写入DM的位宽 |
| 15 | sign\_l | O | 写入DM的值是否进行有符号扩展 |
| 16 | M\_WDSel | O | M级中写入GRF值的选择信号 |
| 17 | D\_rs\_Tuse | O | D级是否读取rs |
| 18 | D\_rt\_Tuse | O | D级是否读取rt |
| 19 | E\_rs\_Tuse | O | E级是否读取rs |
| 20 | E\_rt\_Tuse | O | E级是否读取rt |

（2）真值表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 端口 | addu | subu | ori | lw | sw | beq | lui |
| func | 100001 | 100011 | n/a | n/a | n/a | n/a | n/a |
| op | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 |
| sign | x | x | 0 | 1 | 1 | x | x |
| Branch | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| MemWrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| RegWrite | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| MemtoReg | 0 | 0 | 0 | 1 | x | x | 0 |
| ALUSrc | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| RegDst | 1 | 1 | 0 | 0 | x | x | 0 |
| ALUControl | 000 | 001 | 011 | 000 | 000 | 001 | 100 |

### （三）重要机制实现方法

#### 1. 跳转

CMP模块在D级生成跳转信号

#### 2. 流水线延迟槽

保证在b指令和j指令来临时，一定会执行后一条指令，根据判断结果，再进行跳转

#### 3. 转发

在hazard统一生成，再传到各个流水线级中

#### 4. 进入中断处理程序

用en和flush进行终端和清洗

## 二、测试方案

### （1）测试代码

ori $a0,$0,0x100

ori $a1,$a0,0x123

lui $a2,456

lui $a3,0xffff

ori $a3,$a3,0xffff

addu $s0,$a0,$a2

addu $s1,$a0,$a3

addu $s4,$a3,$a3

subu $s2,$a0,$a2

subu $s3,$a0,$a3

sw $a0,0($0)

sw $a1,4($0)

sw $a2,8($0)

sw $a3,12($0)

sw $s0,16($0)

sw $s1,20($0)

sw $s2,24($0)

sw $s3,44($0)

sw $s4,48($0)

lw $a0,0($0)

lw $a1,12($0)

sw $a0,28($0)

sw $a1,32($0)

ori $a0,$0,1

ori $a1,$0,2

ori $a2,$0,1

beq $a0,$a1,loop1

beq $a0,$a2,loop2

loop1: sw $a0,36($t0)

loop2: sw $a1,40($t0)

jal loop3

jal loop3

sw $s5,64($t0)

ori $a1,$a1,4

jal loop4

loop3:sw $a1,56($t0)

sw $ra,60($t0)

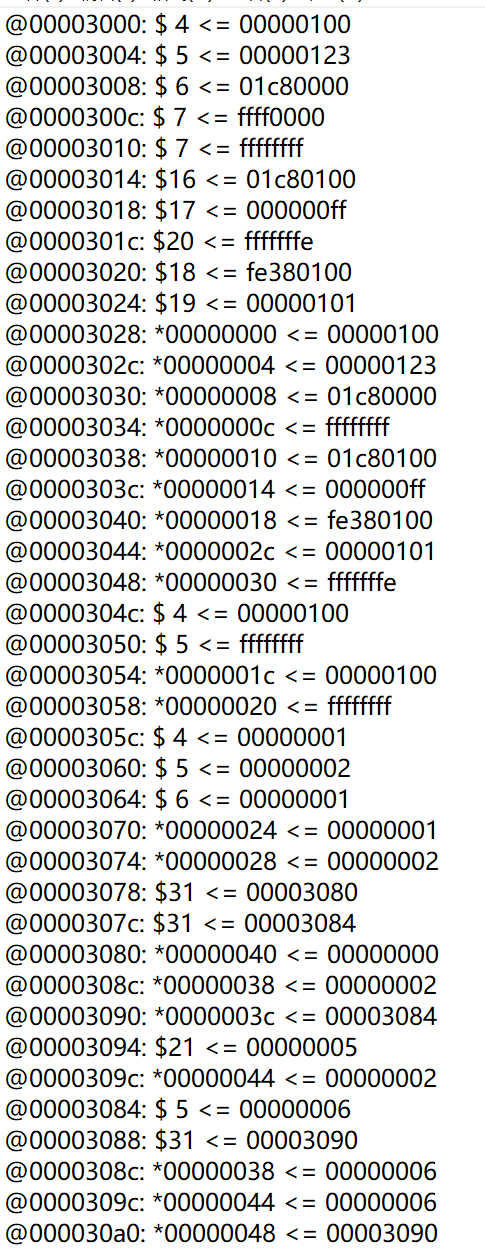
ori $s5,$s5,5

jr $ra

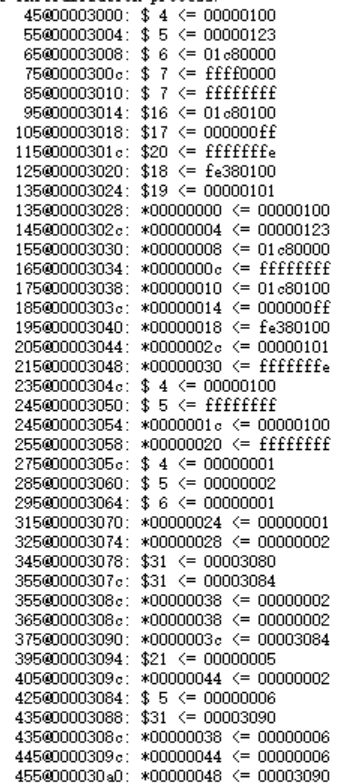
loop4: sw $a1,68($t0)

sw $ra,72($t0)

### （2）MARS中结果



### （3）该CPU中的结果



## 思考题

**（一）为什么需要有单独的乘除法部件而不是整合进ALU？为何需要有独立的HI、LO寄存器？**

因为要遵循“高内聚，低耦合”的设计思想。MD有和ALU不同的特性，是个时钟原件。有独立的HI和LO，操作会更加灵活，多加2个寄存器，得到更大的优势。

**（二）参照你对延迟槽的理解，试解释“乘除槽”。**

把本应暂停的周期利用了。

**（三）举例说明并分析何时按字节访问内存相对于按字访问内存性能上更有优势。**

字符串读写、更改；因为一个字符的ASCII码刚好是一个字节，不用读完整的一个字。

**（四）在本实验中你遇到了哪些不同指令类型组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？**

主要是使用MD的指令之间的冲突。

在D级检测是否该指令要使用MD，暂停的条件是要使用MD并且start或busy处于置一的状态。

测试样例：

乘除运算后面跟上各种要使用MD的指令。