P6设计文档

——Verilog流水线（50指令）

1. 指令集

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| add | addu | sub | subu | sll | srl | sra | sllv | srlv | srav |
| and | or | xor | nor | slt | sltu | addi | addiu | andi | ori |
| xori | lui | slti | sltiu | beq | bne | blez | bgtz | bltz | bgez |
| j | jal | jalr | jr | lb | lbu | lh | lhu | lw | sb |
| sh | sw | mult | multu | div | divu | mfhi | mflo | mthi | mtlo |

注：所有运算类指令均不考虑因溢出而产生的异常。

1. 模块规格
2. PC：指令地址寄存器（F级部件）

模块端口定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 描述 |
| clk |  | I | 内置时钟信号 |
| reset |  | I | 同步复位信号 |
| en |  | I | 写使能信号 |
| PCI | [31:0] | I | 下一条指令的地址 |
| PC | [31:0] | O | 当前指令的地址 |

模块功能说明如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名 | 描述 |
| 1 | 读指令地址 | 通过PC端口输出当前指令地址 |
| 2 | 写指令地址 | 时钟上升沿到来时若en信号为1，则更新指令地址 |
| 3 | 同步复位 | 时钟上升沿到来时若reset信号为1，则复位指令地址至初始状态0x00003000 |

1. IM：指令存储器（F级部件）

模块端口定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 描述 |
| IMI | [31:0] | I | 当前指令地址信号 |
| IM | [31:0] | O | 当前指令信号 |

模块功能说明如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名 | 描述 |
| 1 | 读指令 | 读出指令存储器中IMI地址对应的指令并通过IM端口输出 |
| 2 | 初始化写指令 | 初始时向指令存储器中读入所有指令 |

1. GRF： 寄存器堆（D级部件，W级部件）

模块端口定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 描述 |
| clk |  | I | 内置时钟信号 |
| reset |  | I | 同步复位信号 |
| PC | [31:0] | I | 当前指令的地址信号 |
| A1 | [4:0] | I | 第一个读寄存器地址输入信号 |
| A2 | [4:0] | I | 第二个读寄存器地址输入信号 |
| A3 | [4:0] | I | 写寄存器地址输入信号 |
| WD | [31:0] | I | 写入数据信号 |
| WE |  | I | 写使能信号 |
| RD1 | [31:0] | O | A1所对应的寄存器的数据信号 |
| RD2 | [31:0] | O | A2所对应的寄存器的数据信号 |

模块功能说明如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名 | 描述 |
| 1 | 读数据 | 读出A1，A2地址对应寄存器数据并通过RD1，RD2端口输出 |
| 2 | 写数据 | 时钟上升沿到来时，若WE信号为1，则向A3地址对应寄存器写入数据WD（0号寄存器不能被写入）并输出一条与PC有关的信息 |
| 3 | 同步复位 | 时钟上升沿到来时，若reset信号为1，则复位所有寄存器至初始状态0x00000000 |

1. EXT：立即数拓展模块（D级部件）

模块端口定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 描述 |
| EXTIMM | [15:0] | I | 参与拓展的立即数信号 |
| EXTOp | [1:0] | I | 拓展方式的选择信号 |
| EXT | [31:0] | O | 拓展完成后的数据信号 |

模块功能说明如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名 | 描述 |
| 1 | 符号拓展 | 若EXTOp==00，将立即数加载至输出信号低位并用其最高位填充输出信号的高16位 |
| 2 | 零拓展 | 若EXTOp==01，将立即数加载至输出信号低位并用0填充输出信号的高16位 |
| 3 | 低位零拓展 | 若EXTOp==10，将立即数加载至输出信号高位并用0填充输出信号的低16位 |

1. NPC：指令地址计算器（D级部件）

模块端口定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 描述 |
| ADD4 | [31:0] | I | PC+4信号 |
| PC4\_D | [31:0] | I | F/D级流水线寄存器中的PC4信号 |
| imm\_index | [25:0] | I | 计算跳转地址的imm/index数据信号 |
| M\_RD1\_D | [31:0] | I | D级读取rs寄存器时的转发信号 |
| CMP |  | I | 跳转时特殊条件的判断信号 |
| Is\_B |  | I | b类指令的判断信号（beq/bne/blez/bgtz/bltz/bgez） |
| Is\_J |  | I | j类指令的判断信号（j / jal） |
| PCOp | [1:0] | I | 计算下一条指令地址的选择信号 |
| TURE\_NPC | [31:0] | O | 下一条指令地址 |

模块功能说明如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名 | 描述 |
| 1 | 计算指令地址 | 若PCOp==00，输出PC+4信号  若PCOp==01，输出M\_RD1\_D信号  若PCOp==10且is\_B==1且CMP==1，则计算PC4\_D+imm拓展并输出  若PCOp==10且is\_J==1，则计算PC4\_D与index拓展信号并输出  若PCOp==10且不属于以上情况，则输出PC4\_D+4信号 |

1. CMP：数据比较器（D级部件）

模块端口定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 描述 |
| D1 | [31:0] | I | 参与比较的第一个数据信号 |
| D2 | [31:0] | I | 参与比较的第二个数据信号 |
| CMPOp | [2:0] | I | 比较方式的选择信号 |
| CMP |  | O | 比较结果信号 |

模块功能说明如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名 | 描述 |
| 1 | 比较相等 | 若CMPOp==000，二数据信号相等，则输出1，否则输出0 |
| 2 | 比较不等 | 若CMPOp==001，若二数据信号不等，则输出1，否则输出0 |
| 3 | 比较小于0 | 若CMPOp==010，若D1小于0，则输出1，否则输出0 |
| 4 | 比较小于等于0 | 若CMPOp==011，若D1小于等于0，则输出1，否则输出0 |
| 5 | 比较大于0 | 若CMPOp==100，若D1大于0，则输出1，否则输出0 |
| 6 | 比较大于等于0 | 若CMPOp==101，若D1大于等于0，则输出1，否则输出0 |

1. ALU：计算模块（E级部件）

模块端口定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 描述 |
| A | [31:0] | I | 参与运算的第一个数据信号 |
| B | [31:0] | I | 参与运算的第二个数据信号 |
| ALUOp | [3:0] | I | 运算方式的选择信号 |
| ALU | [31:0] | O | 运算结果的数据信号 |

模块功能说明如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名 | 描述 |
| 1 | 按位与运算 | 若ALUOp==0000，计算A&B并通过ALU端口输出 |
| 2 | 按位或运算 | 若ALUOp==0001，计算A|B并通过ALU端口输出 |
| 3 | 不支持溢出加法 | 若ALUOp==0010，计算A+B并通过ALU端口输出 |
| 4 | 不支持溢出减法 | 若ALUOp==0011，计算A-B并通过ALU端口输出 |
| 5 | 逻辑左移 | 若ALUOp==0100，计算B逻辑左移A并通过ALU端口输出 |
| 6 | 逻辑右移 | 若ALUOp==0101，计算B逻辑右移A并通过ALU端口输出 |
| 7 | 算术右移 | 若ALUOp==0110，计算B算术右移A并通过ALU端口输出 |
| 8 | 异或 | 若ALUOp==0111，计算A xor B并通过ALU端口输出 |
| 9 | 或非 | 若ALUOp==1000，计算A nor B并通过ALU端口输出 |
| 10 | 小于置1 | 若ALUOp==1001，A < B则输出1，否则输出0 |
| 11 | 无符号小于置1 | 若ALUOp==1010，A无符号 < B则输出1，否则输出0 |

1. XALU：乘除法计算模块（E级部件）

模块端口定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 描述 |
| clk |  | I | 内置时钟信号 |
| reset |  | I | 同步复位信号 |
| XALUA | [31:0] | I | 参与运算的第一个数据信号 |
| XALUB | [31:0] | I | 参与运算的第二个数据信号 |
| XALUOp | [2:0] | I | 运算方式的选择信号 |
| START |  | I | 乘除法计算的开始信号 |
| BUSY |  | O | 正在运算乘除法的判断信号 |
| HIGH | [31:0] | O | HI寄存器的值 |
| LOW | [31:0] | O | LO寄存器的值 |

模块功能说明如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名 | 描述 |
| 1 | 同步复位 | 时钟上升沿到来时，若reset信号为1，则复位HI和LO至0x00000000，复位BUSY至0 |
| 2 | 符号乘法 | 时钟上升沿到来时，若START信号为1且XALUOp==000，则计算A\*B并在5个时钟周期后通过HIGH和LOW输出 |
| 3 | 无符号乘法 | 时钟上升沿到来时，若START信号为1且XALUOp==001，则计算无符号A\*B并在5个时钟周期后通过HIGH和LOW输出 |
| 4 | 符号除法 | 时钟上升沿到来时，若START信号为1且XALUOp==010，则计算A/B并在10个时钟周期后通过HIGH和LOW输出 |
| 5 | 无符号除法 | 时钟上升沿到来时，若START信号为1且XALUOp==011，则计算无符号A/B并在10个时钟周期后通过HIGH和LOW输出 |
| 6 | 写HI寄存器 | 时钟上升沿到来时，若XALUOp==100，则将XALUA值写入HI |
| 8 | 写LO寄存器 | 时钟上升沿到来时，若XALUOp==100，则将XALUA值写入LO |

1. DM：数据存储器（M级部件）

模块端口定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 描述 |
| clk |  | I | 内置时钟信号 |
| reset |  | I | 同步复位信号 |
| PC | [31:0] | I | 当前指令的地址信号 |
| DMA | [31:0] | I | 存取的地址信号 |
| DMD | [31:0] | I | 存取的数据信号 |
| DMWE |  | I | 存数据使能信号 |
| DMOp | [1:0] | I | 存数据方式的选择信号 |
| DM | [31:0] | O | 取出的数据信号 |

模块功能说明如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名 | 描述 |
| 1 | 存数据 | 时钟上升沿到来时，若DMWE信号为1，则根据DMOp的值向数据存储器DMA对应地址的某一段中写入DMD数据并输出一条与PC有关的信息 |
| 2 | 取数据 | 将数据存储器DMA地址对应的数据通过DM端口输出 |
| 3 | 同步复位 | 时钟上升沿到来时，若reset信号为1，则复位数据存储器至初始状态0x00000000 |

1. DMEXT：数据存储器拓展器（M级部件）

模块端口定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 描述 |
| DM | [31:0] | I | 所需要拓展的数据信号 |
| DMA | [31:0] | I | 读取数据存储器的地址信号 |
| DMEXTOp | [2:0] | I | 拓展方式选择信号 |
| DMEXT | [31:0] | O | 拓展后的数据信号 |

模块功能说明如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名 | 描述 |
| 1 | 拓展数据 | 根据DMA末两位值与DMEXTOp值对DM数据进行拓展后通过DMEXT端口输出 |

1. 流水线寄存器规格
2. FDreg：F/D级流水线寄存器

模块端口定义如下：

|  |  |  |
| --- | --- | --- |
| 控制信号 | 输入信号 | 输出信号 |
| clk | PC | IR\_D |
| reset | IM | PC\_D |
| en |  | PC4\_D |
|  |  | PC8\_D |

1. DEreg：D/E级流水线寄存器

模块端口定义如下：

|  |  |  |
| --- | --- | --- |
| 控制信号 | 输入信号 | 输出信号 |
| clk | IR\_D | IR\_E |
| reset | PC\_D | PC\_E |
| en | PC4\_D | PC4\_E |
|  | PC8\_D | PC8\_E |
|  | RD1 | RD1\_E |
|  | RD2 | RD2\_E |
|  | EXT | EXT\_E |
|  | GRFWE | GRFWE\_E |
|  | Tnew | Tnew\_E |

1. EMreg：E/M级流水线寄存器

模块端口定义如下：

|  |  |  |
| --- | --- | --- |
| 控制信号 | 输入信号 | 输出信号 |
| clk | IR\_E | IR\_M |
| reset | PC\_E | PC\_M |
| en | PC4\_E | PC4\_M |
|  | PC8\_E | PC8\_M |
|  | ALU | ALU\_M |
|  | RD2\_E | RD2\_M |
|  | GRFWE\_E | GRFWE\_M |
|  | Tnew\_E | Tnew\_M |

1. MWreg：M/W级流水线寄存器

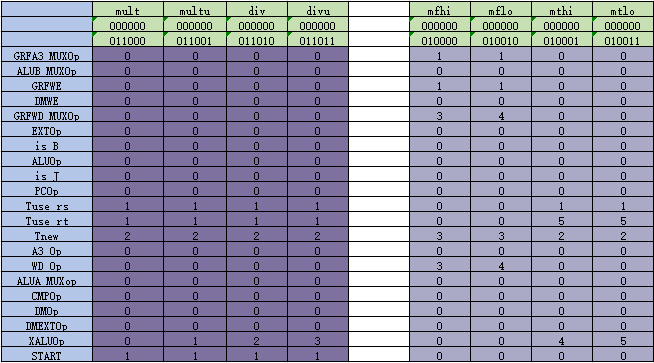
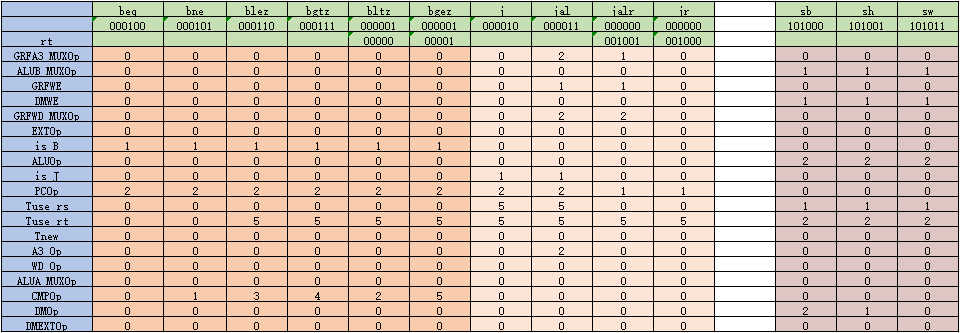
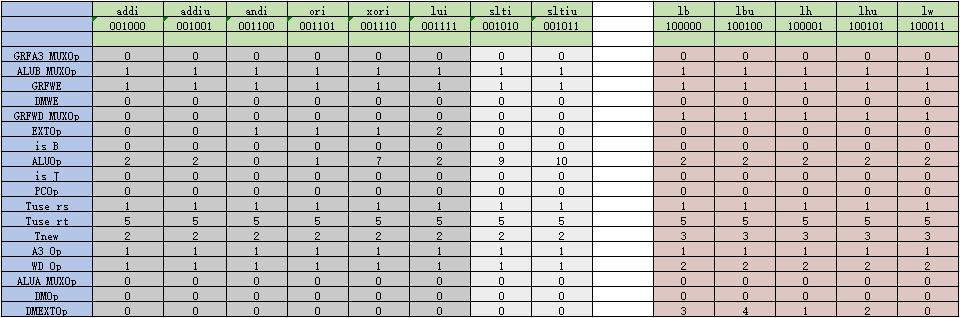
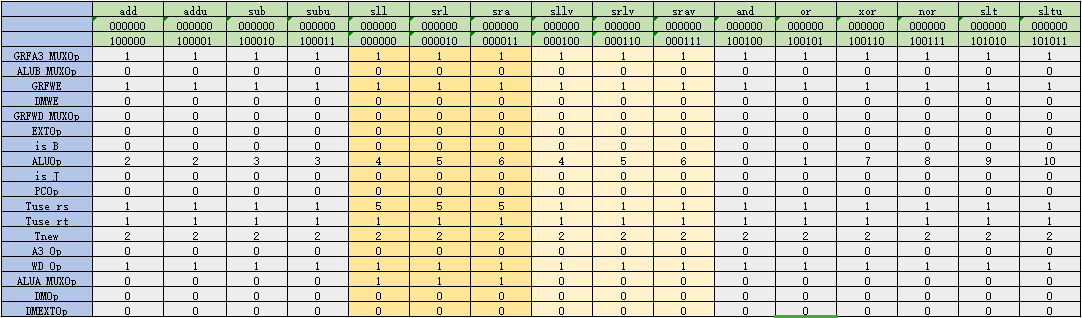
模块端口定义如下：

|  |  |  |
| --- | --- | --- |
| 控制信号 | 输入信号 | 输出信号 |
| clk | IR\_M | IR\_W |
| reset | PC\_M | PC\_W |
| en | PC4\_M | PC4\_W |
|  | PC8\_M | PC8\_W |
|  | ALU\_M | ALU\_W |
|  | DM | DM\_W |
|  | GRFWE\_M | GRFWE\_W |

1. 控制器设计
2. 控制信号说明



1. 指令与控制信号真值表



1. 冒险处理策略

在P6中共有50条指令，仍使用P5中所采取的分类表格法将十分繁杂且易于出错，因此，本设计采取暴力转发的方法。其主要实现方式为，在每一级流水线寄存器中都存入当前指令的Tnew（距离产生结果所剩的时钟周期数）以及A3（所将要写入的5位寄存器地址）和WD（所将要写入的数据），同时D级控制器产生D级指令的Tuse\_rs（距需要使用RD1所剩的时钟周期数）和Tuse\_rt（距需要使用RD2所剩的时钟周期数）。每一次时钟上升沿到来时，若检测到D级指令需要用到的RD1或RD2仍未产生，则产生一次暂停，若检测到其他已产生结果的数据冒险，则进行转发。这种设计保证了每一条指令所使用的值均是正确的。此外，乘除法的暂停也由BUSY信号决定。

有关模块：

1. Analysis\_MUX（A3及WD分析模块）

功能：根据每一级控制器产生的A3\_MUX和WD\_MUX信号产生每一级指令的A3与WD

1. StopUnit（暂停分析与产生模块）

功能：检测到D级指令所需数据不能及时得出时进行一次暂停，D级为乘除法相关指令且XALU忙碌时进行一次暂停

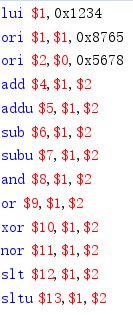
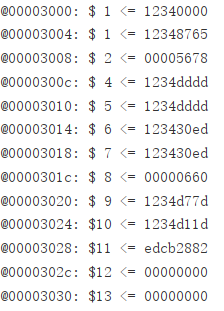
1. ForwardUnit（转发来源分析与产生模块）

功能：在D级rs、D级rt、E级rs、E级rt、M级rt五个位点检测到数据冒险时根据各个流水线寄存器的WD和A3值进行转发

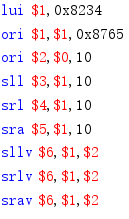
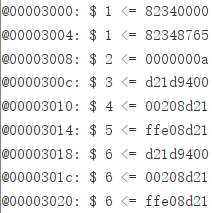
1. 测试样例

首先对指令本身行为进行测试，测试程序与期望结果如下

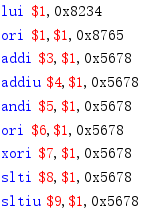
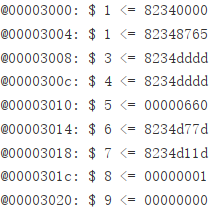
1. add addu sub subu and or xor nor slt sltu

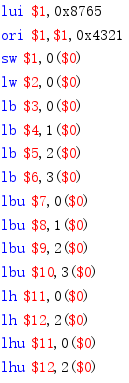
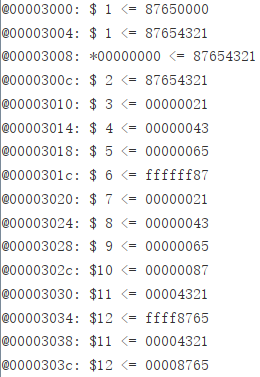
1. Sll srl sra sllv srlv srav

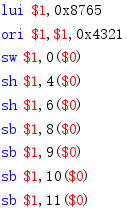
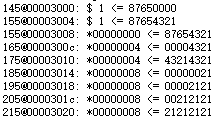
1. Addi addiu andi ori xori lui slti sltiu

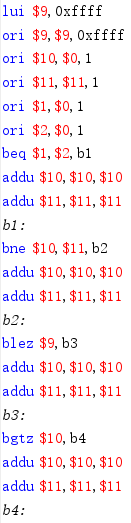
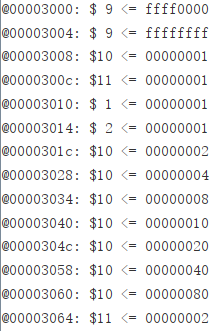
1. Lb lbu lh lhu lw

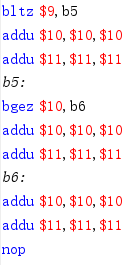
 

1. Sb sh sw

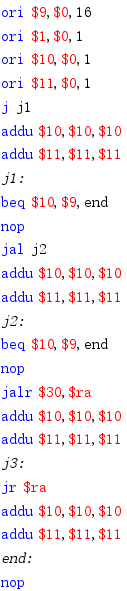
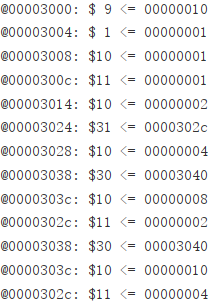
 

1. Beq bne blez bgtz bltz bgez

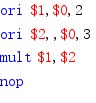
 



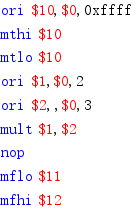
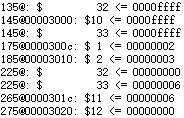
1. J jal jalr jr

1. Mult multu div divu（测试此类指令时打开写入HILO时的输出信息）

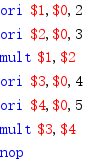
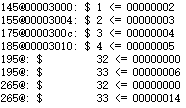
 fe70c3bc051f9ed6b7465371fb16202

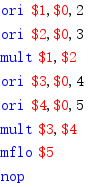
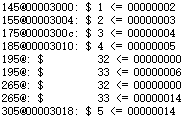
1. Mfhi mflo mthi mtlo（测试此类指令时打开写入HILO时的输出信息）

确认指令本身行为无误后，进行暂停与转发相关的测试，由于使用暴力转发的方法且P5所使用指令集已确保正确，因此在P6中加入calr、cali、b、j、load、store类型指令时均可认为暂停与转发正确，因此测试的重点为乘除法指令之间的冲突与乘除法指令和其他类型指令之间的冲突。

乘除法指令之间的冲突：（需要观察波形图以确定暂停周期数）

乘除法指令与其他指令之间的冲突：

至少应在每类指令中均选取一个进行冲突测试，以下为例：

9c2fc396b9f2444053035fbf432fe2b

思考题

1. 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？

每条乘法指令需要5个周期，每条除法指令需要10个周期，若将乘除法也整合进ALU，则每次出现乘除法指令时整条流水线都需要暂停5或10个周期（因不需要使用ALU的指令极少），会造成效率的巨大损失，而单独使用XALU后乘除法指令后的非乘除法指令不会被阻塞，对流水线效率影响较小。乘除法结果为64位而其他一般为32位，故需要独立的两个寄存器用于存储结果，这样能减少流水线寄存器等的连线和复杂程度。

1. 参照你对延迟槽的理解，试解释“乘除槽”。

当乘除法正在计算时XALU的BUSY端口输出为1，此时非乘除法指令仍能正确通过流水线传递而不受影响，此即为在“乘除槽”内。正是因为有“乘除槽”的存在，流水线才能够在E级才开始分析指令类型而不需要在D级便分析完全。

1. 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。（Hint：考虑C语言中字符串的情况）

当汇编语言中连续出现lb指令而lw指令较少时（如C语言中在字符串中取字符的情况）按字节访问内存相对于按字访问性能上更有优势，因每个char类型变量仅占据1byte内存，按字节访问后不需对所取数据进行额外的拓展处理。

1. 如何概括你所设计的CPU的设计风格？为了对抗复杂性你采取了哪些抽象和规范手段？

我的暂停与转发模块为DETECTOR型，它接收各流水线寄存器上指令的WD和A3和Tnew信息，在发生数据冒险时及时进行暂停或转发。为了对抗复杂性，我的暂停与转发模块由多个小模块构成，Analysis\_MUX用于由各级指令本身分析出各级WD与A3，StopUnit用于由各级A3与TnewTuse产生暂停信号，ForwardUnit用于数据转发，并一并连接到数据通路中，这样每个模块的代码量不会过多，同时功能简洁，在出现bug时易于定位错误位置与调试。此外，规范变量名，对齐代码等方法也能够对抗其复杂性。

1. 你对流水线CPU设计风格有何见解？

流水线CPU的各功能部件（PC、ALU、EXT、GRF等）由于其功能的高度特定化而大同小异，因此区分CPU设计风格的主要要点在于控制器的设计和转发暂停的方式。其中，控制器既可以将指令用与或门进行连接并产生控制信号，也可以使用case语句将每一个指令的控制信号单独列出，前者优点在于代码量小，但是难以在调试时看清每一条指令的控制信号，后者代码量大，但可以复制粘贴得出且较清晰，加指令时也较为方便。转发暂停的方式主要有表格法、暴力转发与标记转发，三者思考难度上逐渐递增，但代码量与修改的难易程度逐渐递减。

1. 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？

见测试样例部分。