P5设计文档

——Verilog流水线

1. 指令集
2. ADDU：不支持溢出的加法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编码 | Opcode  000000 | rs | rt | rd | Shamt  00000 | Func  100001 |
| 6 | 5 | 5 | 5 | 5 | 6 |
| 描述 | GPR[rd] ← GPR[rs] + GPR[rt] | | | | | |

1. SUBU：不支持溢出的减法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编码 | Opcode  000000 | rs | rt | rd | Shamt  00000 | Func  100011 |
| 6 | 5 | 5 | 5 | 5 | 6 |
| 描述 | GPR[rd] ← GPR[rs] - GPR[rt] | | | | | |

1. ORI：或立即数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编码 | Opcode  001101 | rs | rt | immediate |
| 6 | 5 | 5 | 16 |
| 描述 | GPR[rt] ← GPR[rs] OR zero\_extend(immediate) | | | |

1. LW：加载字

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编码 | Opcode  100011 | base | rt | offset |
| 6 | 5 | 5 | 16 |
| 描述 | Addr ← GPR[base] + sign\_extend(offset)  GPR[rt] ← memory[Addr] | | | |

1. SW：存储字

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编码 | Opcode  101011 | base | rt | offset |
| 6 | 5 | 5 | 16 |
| 描述 | Addr ← GPR[base] + sign\_extend(offset)  memory[Addr] ← GPR[rt] | | | |

1. BEQ：相等时跳转

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编码 | Opcode  000100 | rs | rt | offset |
| 6 | 5 | 5 | 16 |
| 描述 | If (GPR[rs] == GPR[rt])  PC ← PC + 4 + sign\_extend(offset||0²)  Else  PC ← PC + 4 | | | |

1. LUI：立即数加载至高位

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编码 | Opcode  001111 | 0  00000 | rt | immediate |
| 6 | 5 | 5 | 16 |
| 描述 | GPR[rt] ← immediate||0^16 | | | |

1. JAL：跳转并链接

|  |  |  |
| --- | --- | --- |
| 编码 | Opcode  000011 | index |
| 6 | 26 |
| 描述 | PC ← PC[31:28]||index||00  GPR[31] ← PC+4 | |

1. J：跳转

|  |  |  |
| --- | --- | --- |
| 编码 | Opcode  000010 | index |
| 6 | 26 |
| 描述 | PC ← PC[31:28]||index||00 | |

1. JR：跳转至寄存器

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编码 | Opcode  000000 | rs | 0 | 0 | Func  001000 |
| 6 | 5 | 10 | 5 | 6 |
| 描述 | PC ← GPR[rs] | | | | |

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 |
| ZERO |  | I | b类跳转时的相等信号 |
| Is\_B |  | I | b类指令的判断信号（beq） |
| 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ZERO==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 |
| CMP |  | O | 比较结果信号 |

模块功能说明如下：

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名 | 描述 |
| 1 | 比较数据 | 若二数据信号相等，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 | [2:0] | 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==000，计算A&B并通过ALU端口输出 |
| 2 | 按位或运算 | 若ALUOp==001，计算A|B并通过ALU端口输出 |
| 3 | 不支持溢出加法 | 若ALUOp==010，计算A+B并通过ALU端口输出 |
| 4 | 不支持溢出减法 | 若ALUOp==011，计算A-B并通过ALU端口输出 |

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 |
| 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，则向数据存储器DMA对应地址中写入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. 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 |

1. EMreg：E/M级流水线寄存器

模块端口定义如下：

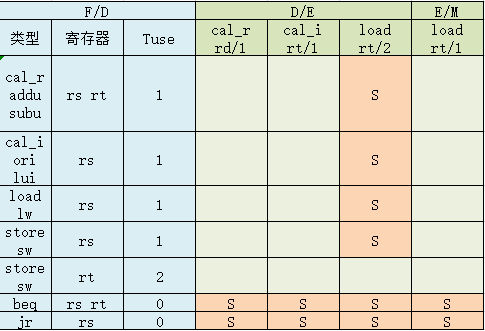
|  |  |  |
| --- | --- | --- |
| 控制信号 | 输入信号 | 输出信号 |
| clk | IR\_E | IR\_M |
| reset | PC\_E | PC\_M |
| en | PC4\_E | PC4\_M |
|  | PC8\_E | PC8\_M |
|  | ALU | ALU\_M |
|  | RD2\_E | RD2\_M |

1. MWreg：M/W级流水线寄存器

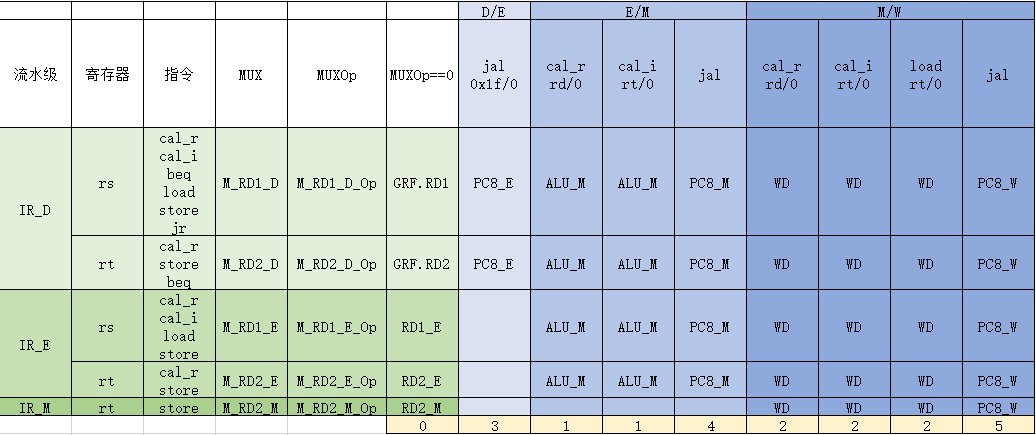
模块端口定义如下：

|  |  |  |
| --- | --- | --- |
| 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 |

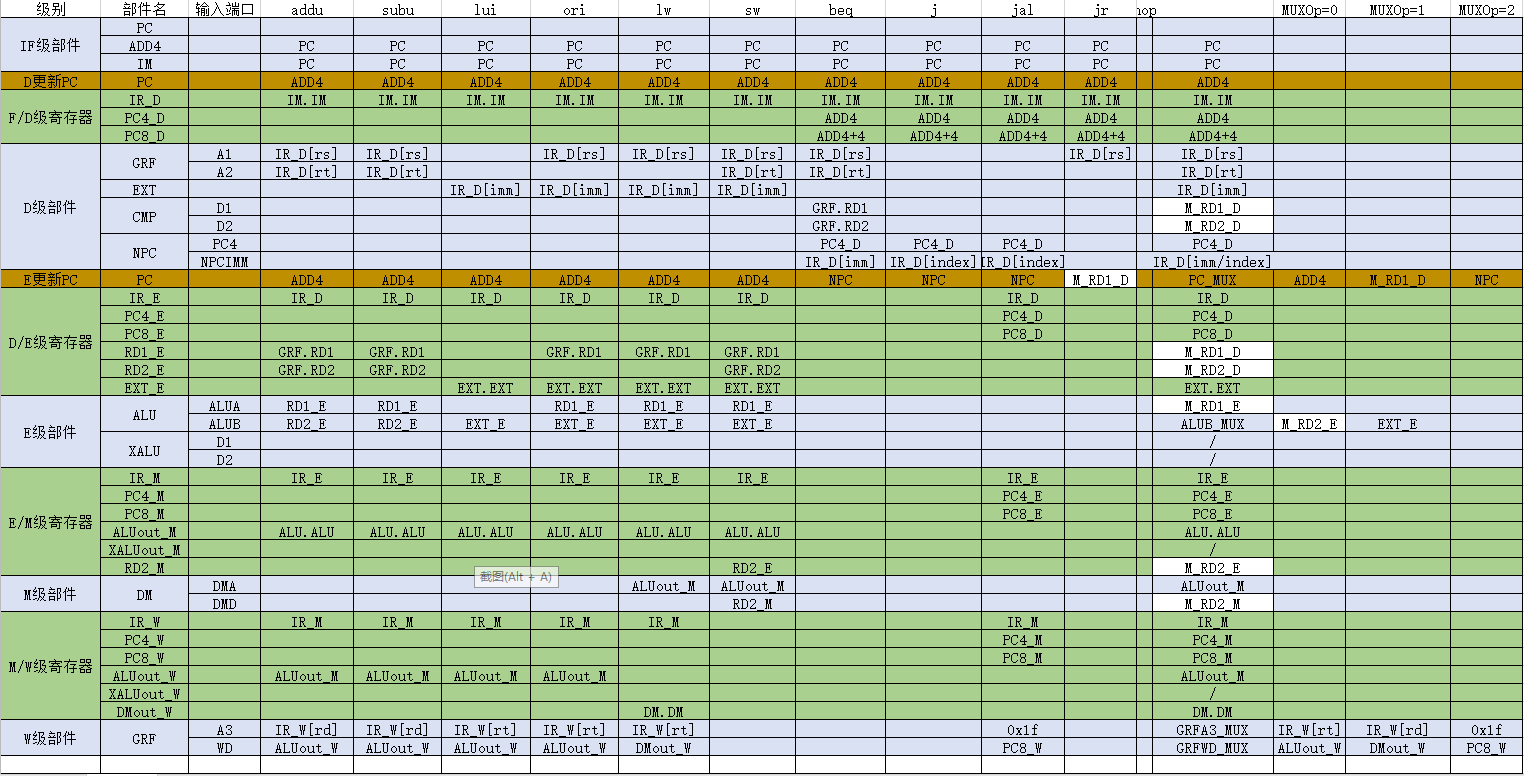
1. 冒险处理策略
2. 暂停处理表



1. 转发处理表



1. 数据通路
2. 数据通路表



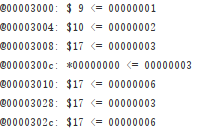
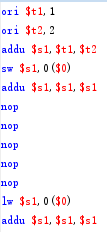
1. 控制器设计
2. 控制信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 位数 | 描述 |
| 1 | ALUOp | [2:0] | ALU计算方式的选择信号 |
| 2 | GRFA3\_MUXOp | [1:0] | GRFA3口连接信号的选择信号 |
| 3 | ALUB\_MUXOp |  | ALUB口连接信号的选择信号 |
| 4 | GRFWE |  | GRF的写使能信号 |
| 5 | DMWE |  | DM的写使能信号 |
| 6 | GRFWD\_MUXOp | [1:0] | GRFWD口连接信号的选择信号 |
| 7 | EXTOp | [1:0] | EXT拓展方式的选择信号 |
| 8 | Is\_B |  | B类指令（beq）的特征信号 |
| 9 | Is\_J |  | J类指令（j/jal）的特征信号 |
| 10 | PCOp | [1:0] | PC计算方式的选择信号 |

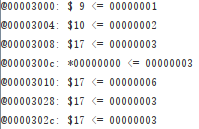
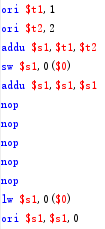
1. 指令与控制信号真值表

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| func | 100001 | 100011 |  |  |  |  |  |  |  | 001000 | 000000 |
| opcode | 000000 | 000000 | 001101 | 001111 | 100011 | 101011 | 000100 | 000010 | 000011 | 000000 | 000000 |
| 指令名 | addu | subu | ori | lui | lw | sw | beq | j | jal | jr | nop |
| GRFA3\_MUXOp | 01 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 10 | 01 | 00 |
| ALUB\_MUXOp | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| GRFWE | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| DMWE | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| GRFWD\_MUXOp | 00 | 00 | 00 | 00 | 01 | 00 | 00 | 00 | 10 | 00 | 00 |
| EXTOp | 00 | 00 | 01 | 10 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| Is\_B | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| ALUOp | 010 | 011 | 001 | 010 | 010 | 010 | 011 | 000 | 000 | 000 | 000 |
| Is\_J | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| PCOp | 00 | 00 | 00 | 00 | 00 | 00 | 010 | 10 | 10 | 01 | 00 |

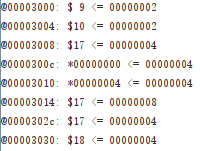
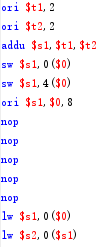
1. 测试样例
2. 暂停组合
3. D: cal\_r E: load



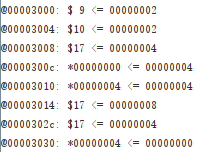
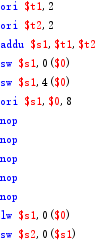
1. D: cal\_i E: load



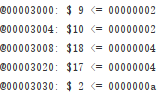
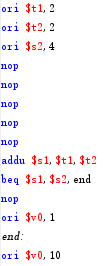
1. D: load E: load



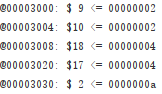
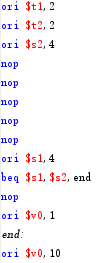
1. D: store E: load



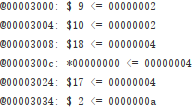
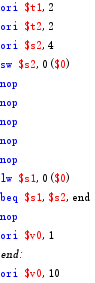
1. D: beq E: cal\_r



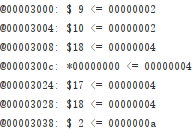
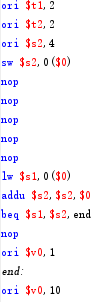
1. D: beq E: cal\_i



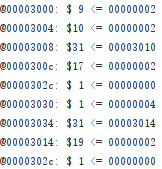
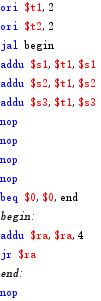
1. D: beq E: load



1. D: beq M: load



1. D: jr E: cal\_r



1. D: jr E: cal\_i

略

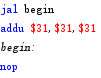
1. D: jr E: load

略

1. D: jr M: load

略

1. 转发组合
2. D: cal\_r E: jal R: rs

8c27a79d44eee59dc814393f222b316

1. D: cal\_i E: jal R: rs

19f45109172cbe1b311bce8345d37e0

1. D: beq E: jal R: rs

Undefined Behaviour

1. D: load E: jal R: rs

略

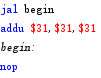
1. D: store E: jal R: rs

略

1. D: jr E: jal R: rs

Undefined Behaviour

1. D: cal\_r E: jal R: rt

8c27a79d44eee59dc814393f222b316

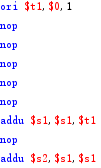
1. D: store E: jal R: rt

略

1. D: beq E: jal R: rt

Undefined Behaviour

1. D: cal\_r M: cal\_r R: rs

2340891d7e0e60b892996da5d145fb2

1. D: cal\_i M: cal\_r R: rs
2. D: beq M: cal\_r R: rs
3. D: load M: cal\_r R: rs
4. D: store M: cal\_r R: rs
5. D: jr M: cal\_r R: rs
6. D: cal\_r M: cal\_r R: rt
7. D: store M: cal\_r R: rt
8. D: beq M: cal\_r R: rt
9. D: cal\_r M: cal\_i R: rs
10. D: cal\_i M: cal\_i R: rs
11. D: beq M: cal\_i R: rs
12. D: load M: cal\_i R: rs
13. D: store M: cal\_i R: rs
14. D: jr M: cal\_i R: rs
15. D: cal\_r M: cal\_i R: rt
16. D: store M: cal\_i R: rt
17. D: beq M: cal\_i R: rt
18. D: cal\_r M: jal R: rs
19. D: cal\_i M: jal R: rs
20. D: beq M: jal R: rs
21. D: load M: jal R: rs
22. D: store M: jal R: rs
23. D: jr M: jal R: rs
24. D: cal\_r M: jal R: rt
25. D: store M: jal R: rt
26. D: beq M: jal R: rt
27. D: cal\_r W: cal\_r R: rs
28. D: cal\_i W: cal\_r R: rs
29. D: beq W: cal\_r R: rs
30. D: load W: cal\_r R: rs
31. D: store W: cal\_r R: rs
32. D: jr W: cal\_r R: rs
33. D: cal\_r W: cal\_r R: rt
34. D: store W: cal\_r R: rt
35. D: beq W: cal\_r R: rt
36. D: cal\_r W: cal\_i R: rs
37. D: cal\_i W: cal\_i R: rs
38. D: beq W: cal\_i R: rs
39. D: load W: cal\_i R: rs
40. D: store W: cal\_i R: rs
41. D: jr W: cal\_i R: rs
42. D: cal\_r W: cal\_i R: rt
43. D: store W: cal\_i R: rt
44. D: beq W: cal\_i R: rt
45. D: cal\_r W: load R: rs
46. D: cal\_i W: load R: rs
47. D: beq W: load R: rs
48. D: load W: load R: rs
49. D: store W: load R: rs
50. D: jr W: load R: rs
51. D: cal\_r W: load R: rt
52. D: store W: load R: rt
53. D: beq W: load R: rt
54. D: cal\_r W: jal R: rs
55. D: cal\_i W: jal R: rs
56. D: beq W: jal R: rs
57. D: load W: jal R: rs
58. D: store W: jal R: rs
59. D: jr W: jal R: rs
60. D: cal\_r W: jal R: rt
61. D: store W: jal R: rt
62. D: beq W: jal R: rt
63. E: cal\_r M: cal\_r R: rs
64. E: cal\_i M: cal\_r R: rs
65. E: load M: cal\_r R: rs
66. E: store M: cal\_r R: rs
67. E: cal\_r M: cal\_r R: rt
68. E: store M: cal\_r R: rt
69. E: cal\_r M: cal\_i R: rs
70. E: cal\_i M: cal\_i R: rs
71. E: load M: cal\_i R: rs
72. E: store M: cal\_i R: rs
73. E: cal\_r M: cal\_i R: rt
74. E: store M: cal\_i R: rt
75. E: cal\_r M: jal R: rs
76. E: cal\_i M: jal R: rs
77. E: load M: jal R: rs
78. E: store M: jal R: rs
79. E: cal\_r M: jal R: rt
80. E: store M: jal R: rt
81. E: cal\_r W: cal\_r R: rs
82. E: cal\_i W: cal\_r R: rs
83. E: load W: cal\_r R: rs
84. E: store W: cal\_r R: rs
85. E: cal\_r W: cal\_r R: rt
86. E: store W: cal\_r R: rt
87. E: cal\_r W: cal\_i R: rs
88. E: cal\_i W: cal\_i R: rs
89. E: load W: cal\_i R: rs
90. E: store W: cal\_i R: rs
91. E: cal\_r W: cal\_i R: rt
92. E: store W: cal\_i R: rt
93. E: cal\_r W: load R: rs
94. E: cal\_i W: load R: rs
95. E: load W: load R: rs
96. E: store W: load R: rs
97. E: cal\_r W: load R: rt
98. E: store W: load R: rt
99. E: cal\_r W: jal R: rs
100. E: cal\_i W: jal R: rs
101. E: load W: jal R: rs
102. E: store W: jal R: rs
103. E: cal\_r W: jal R: rt
104. E: store W: jal R: rt
105. M: store W: cal\_r R: rt
106. M: store W: cal\_i R: rt
107. M: store W: load R: rt
108. M: store W: jal R: rt

思考题

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？

本实验中指令根据其所需要读写的寄存器位置（rs/rt）的不同可分为以下八类：cal\_r（addu/subu）、cal\_i（ori/lui）、store（sw）、load（lw）、beq、j、jal、jr。在指令需要用到RD1/RD2的值时，若此时该数据未完成回写/未产生，则发生了冲突，分别需要用转发/暂停来解决。具体冲突组合见设计文档中的冒险处理策略部分，相应测试样例见设计文档中的测试样例部分。