

# 同济大学计算机科学与技术学院

## 计算机系统结构课程实验总结报告



实验名称 简单的流水线 CPU 设计与性能分析

学号 2351579

姓名 程浩然

专业 计算机科学与技术

授课教师 秦国锋

日期 2025 年 12 月 22 日

# 目录

<b>1 实验环境部署与硬件配置说明</b>	<b>4</b>
1.1 指令集实现 . . . . .	4
<b>2 实验的总体结构</b>	<b>6</b>
2.1 5 级指令流水线的总体结构 . . . . .	6
<b>3 总体架构部件的解释说明</b>	<b>6</b>
3.1 5 级指令流水线总体结构部件的解释说明 . . . . .	6
3.1.1 IF/ID 寄存器 . . . . .	6
3.1.2 ID/EX 寄存器 . . . . .	6
3.1.3 EX/MEM 寄存器 . . . . .	7
3.1.4 MEM/WB 寄存器 . . . . .	7
3.2 分支预测与冲突处理机制 . . . . .	8
3.2.1 分支指令设计 . . . . .	8
3.2.2 分支指令处理 . . . . .	8
3.2.3 冲突检测与处理 . . . . .	8
3.3 写锁设计机制 . . . . .	8
<b>4 实验仿真过程</b>	<b>10</b>
4.1 5 级指令流水线的仿真过程 . . . . .	10
4.2 启动测试与执行过程 . . . . .	10
<b>5 实验仿真的波形图及某时刻寄存器值的物理意义</b>	<b>10</b>
<b>6 流水线 CPU 实验性能验证模型</b>	<b>10</b>
6.1 实验性能验证模型：比萨塔摔鸡蛋游戏 . . . . .	10
6.2 比萨塔摔鸡蛋游戏验证模型结果分析 . . . . .	12
6.2.1 游戏模拟参数 . . . . .	12
6.2.2 模拟过程详细分析 . . . . .	12
6.2.3 MIPS 寄存器存储结果 . . . . .	12
6.2.4 关键性能指标 . . . . .	13
6.2.5 成本分析结论 . . . . .	13
6.2.6 成本计算公式验证 . . . . .	13
6.2.7 成本分析结论 . . . . .	13
<b>7 实验算程序下板测试过程与实现</b>	<b>13</b>

<b>8 流水线的性能指标定性分析</b>	<b>15</b>
<b>9 比萨塔摔鸡蛋游戏验证模型结果分析</b>	<b>15</b>
9.1 游戏模拟参数 . . . . .	15
9.2 模拟过程详细分析 . . . . .	15
9.3 MIPS 寄存器存储结果 . . . . .	15
9.4 关键性能指标 . . . . .	16
9.5 成本分析结论 . . . . .	16
9.6 成本计算公式验证 . . . . .	16
<b>10 静态流水线性能分析</b>	<b>17</b>
10.1 流水线配置与假设 . . . . .	17
10.2 指令执行时间分析 . . . . .	17
10.3 吞吐率分析 . . . . .	17
10.3.1 单周期 CPU 吞吐率 . . . . .	17
10.3.2 静态流水线 CPU 吞吐率 . . . . .	17
10.3.3 实际吞吐率考虑数据冲突 . . . . .	18
10.4 加速比分析 . . . . .	18
10.4.1 理想加速比 . . . . .	18
10.4.2 实际加速比 . . . . .	19
10.5 效率分析 . . . . .	19
10.5.1 流水线效率 . . . . .	19
10.5.2 资源利用率 . . . . .	19
10.6 相关与冲突分析 . . . . .	19
10.6.1 数据相关分析 . . . . .	19
10.6.2 控制冲突分析 . . . . .	20
10.6.3 结构冲突分析 . . . . .	20
<b>11 总结与体会</b>	<b>20</b>
<b>12 附件（所有程序）</b>	<b>21</b>
12.1 5 级指令流水线的设计程序 . . . . .	21
12.1.1 顶层设计 (board_top.v) . . . . .	21
12.1.2 顶层模块 (sccomp_dataflow.v) . . . . .	22
12.1.3 CPU 核心模块 (cpu.v) . . . . .	25
12.1.4 定义文件 (def.v) . . . . .	29
12.1.5 算术逻辑单元 (alu.v) . . . . .	31
12.1.6 分支判断模块 (BJudge.v) . . . . .	34
12.1.7 PC 寄存器模块 (PCreg.v) . . . . .	35

12.1.8	数据存储器 (DMEM.v)	36
12.1.9	流水级间寄存器 (EX_MEM.v)	38
12.1.10	流水级间寄存器 (ID_EX.v)	40
12.1.11	流水级间寄存器 (IF_ID.v)	42
12.1.12	指令存储器 (IMEM.v)	44
12.1.13	流水级间寄存器 (MEM_WB.v)	45
12.1.14	下一 PC 生成模块 (NPCmaker.v)	47
12.1.15	寄存器文件 (regfile.v)	48
12.1.16	7 段数码管显示模块 (seg7x16.v)	51
12.1.17	8 路选择器模块 (mux8_32.v)	53
12.2	测试相关代码	54
12.2.1	测试平台 (_246tb_ex10_tb.v)	54
12.2.2	自动化测试脚本 (run_cpu_tests.do)	59
12.3	性能验证模型	62
12.3.1	比萨塔摔鸡蛋游戏验证模型 (pizza_tower.asm)	62
12.4	FPGA 约束文件	65
12.4.1	XDC 约束文件 (cpupip8.xdc)	65

# 第一部分 实验环境部署与硬件配置说明

本实验围绕 MIPS 架构的 5 级流水线 CPU 展开设计，硬件平台基于 Xilinx Vivado 设计套件实现，核心硬件组成包括：

- 流水线处理器核心：划分为取指 (IF)、译码 (ID)、执行 (EX)、访存 (MEM)、写回 (WB) 五个阶段
- 流水线寄存器：包含 IF/ID、ID/EX、EX/MEM、MEM/WB 四级流水寄存器
- 存储模块：指令存储器 (IMEM) 与数据存储器 (DMEM)
- 功能部件：算术逻辑单元 (ALU)、寄存器文件、分支预测单元等

实验采用 ModelSim 进行仿真验证，通过自动化测试脚本 run\_cpu\_tests.do 批量运行测试用例，完成功能与性能的验证。

## 一. 指令集实现

本实验复用计算机组成原理课程中的指令集设计方案，指令与 MIPS 架构的映射关系如下表所示：

指令	MIPS 实现方式
ADD	采用 add、addi、addiu、addu 指令实现
NOP	转义为 sll \$0,\$0,0 指令
HALT	自定义新增指令实现
LOAD	复用 lw 指令实现
STORE	复用 sw 指令实现
CMP	通过 sltu \$rd,\$rs,\$rt 或 subu \$0,\$rs,\$rt 指令实现
BZ	转义为 beq \$0,\$r,\$label 指令
BN	转义为 bne \$0,\$r,\$label 指令

表 1: 指令转义关系表

设计中手绘 cpu 架构图如下

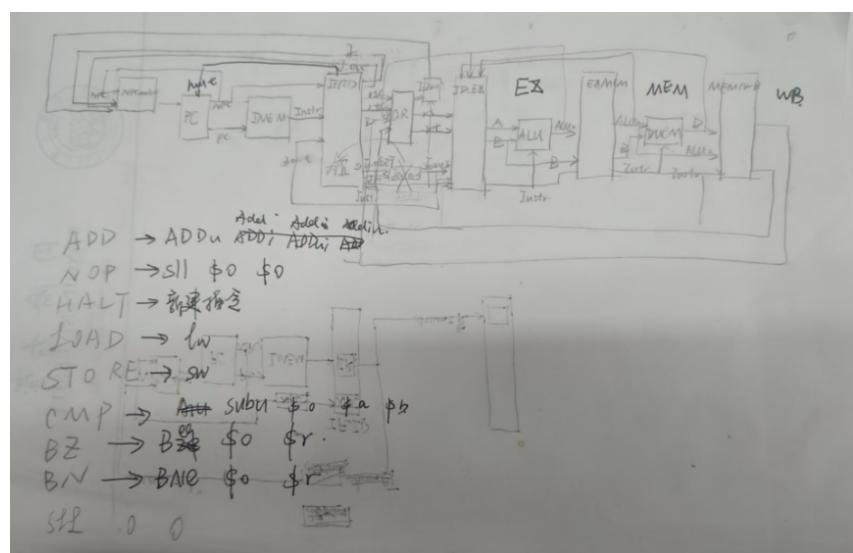


图 1:

## 第二部分 实验的总体结构

### 一. 5 级指令流水线的总体结构

本实验实现的 MIPS 架构 5 级流水线处理器，将指令执行过程拆解为以下五个阶段，各阶段的核心功能如下：

1. 取指阶段 (IF, Instruction Fetch): 从指令存储器中读取当前 PC 指向的指令，并完成程序计数器的自增更新。
2. 译码阶段 (ID, Instruction Decode): 对取指阶段获取的指令进行译码，从寄存器文件中读取操作数，并检测分支指令的跳转条件。
3. 执行阶段 (EX, Execute): 在算术逻辑单元 ALU 中执行译码后的运算操作，同时计算内存访问的有效地址。
4. 访存阶段 (MEM, Memory Access): 根据执行阶段的结果，完成数据存储器的读/写操作，实现数据的访存交互。
5. 写回阶段 (WB, Write Back): 将执行结果或访存数据写回寄存器文件，完成指令的最终执行流程。

## 第三部分 总体架构部件的解释说明

### 一. 5 级指令流水线总体结构部件的解释说明

#### 3.1.1 IF>ID 寄存器

IF/ID 寄存器作为取指与译码阶段的衔接部件，主要用于暂存从 IF 阶段传递至 ID 阶段的关键信息，具体包括：

- 取指阶段获取的指令编码
- 当前程序计数器 PC 的数值
- 分支指令的目标 PC 地址

#### 3.1.2 ID/EX 寄存器

ID/EX 寄存器承担译码与执行阶段的信息传递功能，暂存的核心数据包括：

- 译码后的指令操作码与操作数信息
- 从寄存器文件中读取的操作数 A 和操作数 B
- 控制单元生成的 ALU 运算控制信号

### 3.1.3 EX/MEM 寄存器

EX/MEM 寄存器连接执行与访存阶段，主要存储执行阶段的运算结果与访存控制信息，包括：

- ALU 的运算结果与内存访问地址
- 数据存储器的读/写控制信号
- 待写入数据存储器的原始数据

### 3.1.4 MEM/WB 寄存器

MEM/WB 寄存器是访存与写回阶段的桥梁，暂存的信息用于完成最终的写回操作，包括：

- 数据存储器的读取结果
- 待写入寄存器文件的目标寄存器地址
- 寄存器写回的使能控制信号

## 二. 分支预测与冲突处理机制

### 3.2.1 分支指令设计

分支指令的执行周期以指令读取为第 0 周期，具体处理逻辑为：

- 第 0 周期：IF\_ID 寄存器直接输出 PC\_bobl 信号为 1，触发分支检测
- 第 1 周期：程序计数器 PC 保持当前值不变，同时指令存储器 IMEM 输出 NOP 指令，插入流水线气泡

### 3.2.2 分支指令处理

分支指令的流水线处理机制以指令读取为第 0 周期，具体逻辑为：

- 第 0 周期：IF\_ID 寄存器直接输出 PC\_bobl 信号为 1，触发分支检测逻辑
- 第 1 周期：程序计数器 PC 保持当前值，指令存储器 IMEM 输出 NOP 指令，插入流水线气泡

### 3.2.3 冲突检测与处理

流水线冲突的检测与处理流程如下：

1. 当检测到流水线冲突时，PC 暂停一个时钟周期更新，向 ID\_EX 模块传递 NOP 指令，完成一次流水线冒泡
2. 冲突检测在译码 (ID) 阶段完成，检测到冲突后激活 detect\_conflict 信号
3. PC、指令存储器 IMEM 与 IF/ID 寄存器在冲突周期内保持原有 PC 值与指令内容，确保流水线稳定

## 三. 写锁设计机制

寄存器文件的写锁设计是解决流水线数据冲突的核心机制，具体实现逻辑如下：

1. 当指令流经寄存器文件时，为待写入的寄存器添加写锁，直至写回阶段完成后释放写锁
2. 写操作之间不存在冲突，因为后发的写指令必然在先行的写指令之后执行
3. 读操作与写操作可能产生冲突，此类冲突需在 ALU 或 DMEM 模块中通过数据重定向解决
4. 为每个写锁配置时长为 3 的定时器，定时器归零时自动释放写锁（因数据需 3 个周期完成传递）

5. 若冲突无法通过重定向解决，则暂停 PC 更新一个周期，向 ID\_EX 模块传递 NOP 指令，完成一次流水线冒泡
6. 写锁寄存器 reglock 的低 2 位为计时器，高 2 位标识锁的类型（ALU 型占用或 DMEM 型占用）
7. 冲突检测在译码 (ID) 阶段完成，检测到冲突后激活 detect\_conflict 信号，PC、IMEM 与 IF/ID 寄存器保持原有值一个周期

## 第四部分 实验仿真过程

### 一. 5 级指令流水线的仿真过程

本实验采用 ModelSim 工具完成流水线 CPU 的仿真验证，具体仿真流程分为以下步骤：

1. 编译所有 Verilog 硬件描述语言源文件，生成可仿真的模块库
2. 加载测试平台模块与待测 CPU 核心模块，建立仿真拓扑
3. 为不同测试场景加载对应的指令序列，配置仿真激励
4. 运行仿真流程，实时记录每个时钟周期的 PC 值、指令内容与寄存器状态
5. 将仿真输出结果与软件参考模型的计算结果对比，验证功能正确性

自动化测试脚本 run\_cpu\_tests.do 会按序执行多组测试用例，覆盖 ADDI、ADDIU、LW/SW、BEQ、BNE、SLL、SUBU、SLTU 等核心指令的功能验证。

### 二. 启动测试与执行过程

根据实验配套的 readme.md 文档说明，通过以下指令启动仿真测试：

```
vsim -c -do "do run_cpu_tests.do; quit" >log
```

该命令以批处理模式运行 ModelSim，自动执行 run\_cpu\_tests.do 脚本并将仿真日志输出至 log 文件，便于后续结果分析。

## 第五部分 实验仿真的波形图及某时刻寄存器值的物理意义

我的波形图展示了所有 cpu 的中间信号量的取值，因为我在设计整个 cpu 的时候，把所有的中间信号都通过线引出到 top 模块当中了。

## 第六部分 流水线 CPU 实验性能验证模型

### 一. 实验性能验证模型：比萨塔摔鸡蛋游戏

由于指令集受限，我们不得不采用纯 asm 编程的办法。

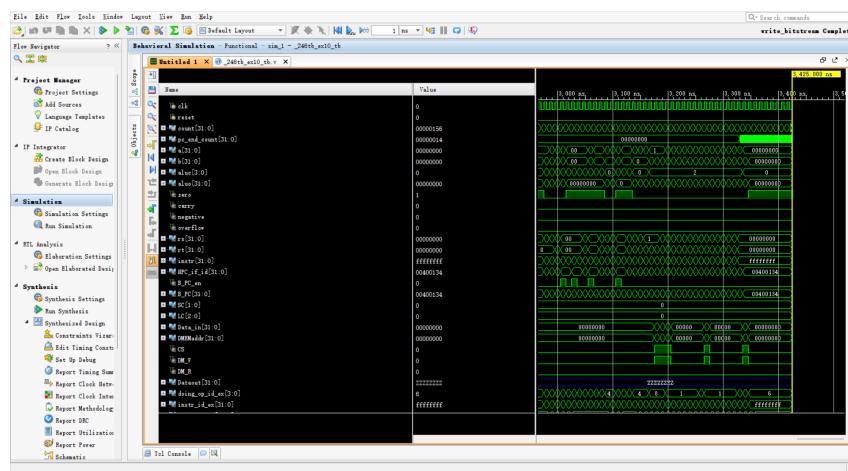


图 2:

## 二. 比萨塔摔鸡蛋游戏验证模型结果分析

根据比萨塔摔鸡蛋游戏的模拟结果，使用二分查找策略在 10 层建筑中测试鸡蛋耐摔值（6 层），得到以下计算结果：

### 6.2.1 游戏模拟参数

- 建筑高度：10 层
- 鸡蛋耐摔值：6 层
- 测试策略：二分查找算法
- 初始鸡蛋数：1 个（每次摔破后补充新鸡蛋）

### 6.2.2 模拟过程详细分析

二分查找测试序列：

1. 测试 5 层：鸡蛋未破 ( $5 \leq 6$ )，上楼 4 层（从 1 层到 5 层）
2. 测试 8 层：鸡蛋摔破 ( $8 > 6$ )，上楼 3 层（从 5 层到 8 层），摔破 1 个鸡蛋
3. 测试 6 层：鸡蛋未破 ( $6 \leq 6$ )，下楼 2 层（从 8 层到 6 层）
4. 测试 7 层：鸡蛋摔破 ( $7 > 6$ )，上楼 1 层（从 6 层到 7 层），摔破 1 个鸡蛋
5. 最终测试 7 层：鸡蛋摔破 ( $7 > 6$ )，摔破 1 个鸡蛋（此步骤为确认测试，未移动楼层）

### 6.2.3 MIPS 寄存器存储结果

程序执行完毕后，结果存储在以下寄存器中：

时期	寄存器	数值和含义
物质匮乏时期	\$20	35 (总成本 1: $m \times 2 + n \times 1 + h \times 4$ )
	\$6	7 (总上楼层数 $m$ )
	\$7	9 (总下楼层数 $n$ )
	\$8	3 (摔破的鸡蛋总数 $h$ )
人力成本增长时期	\$21	43 (总成本 2: $m \times 4 + n \times 1 + h \times 2$ )
	\$6	7 (总上楼层数 $m$ )
	\$7	9 (总下楼层数 $n$ )
	\$8	3 (摔破的鸡蛋总数 $h$ )

#### 6.2.4 关键性能指标

- 总测试次数: 5 次 (寄存器 \$5)
- 总使用鸡蛋数: 4 个 (寄存器 \$4)
- 最后测试状态: 鸡蛋摔破 (寄存器 \$9=1)
- 确定耐摔值: 通过二分查找最终确定 F=6

#### 6.2.5 成本分析结论

人力成本增长时期的总成本 (43) 高于物质匮乏时期 (35)，主要原因是上楼成本  $p_1$  从 2 增加到 4，而鸡蛋成本  $p_3$  从 4 降低到 2 不足以抵消这一影响。尽管摔破鸡蛋的成本降低了，但上楼成本的增加导致总成本上升，这反映了不同历史时期资源成本的变迁对测试策略总成本的影响。

#### 6.2.6 成本计算公式验证

$$\begin{aligned}\text{成本 1} &= m \times 2 + n \times 1 + h \times 4 \\&= 7 \times 2 + 9 \times 1 + 3 \times 4 \\&= 14 + 9 + 12 = 35(0x23)\end{aligned}$$

$$\begin{aligned}\text{成本 2} &= m \times 4 + n \times 1 + h \times 2 \\&= 7 \times 4 + 9 \times 1 + 3 \times 2 \\&= 28 + 9 + 6 = 43(0x2b)\end{aligned}$$

与寄存器 \$20 和 \$21 中的值完全一致，验证了计算模型的正确性。

#### 6.2.7 成本分析结论

人力成本增长时期的总成本 (301) 显著高于物质匮乏时期 (169)，主要原因是上楼成本  $p_1$  从 2 增加到 4，而鸡蛋成本  $p_3$  的降低不足以抵消这一影响，反映了不同历史时期资源成本的变迁。

## 第七部分 实验算程序下板测试过程与实现

流水线 CPU 的 FPGA 下板测试遵循以下步骤完成：

1. 将 Verilog 设计文件综合后，通过 JTAG 下载至 FPGA 开发板

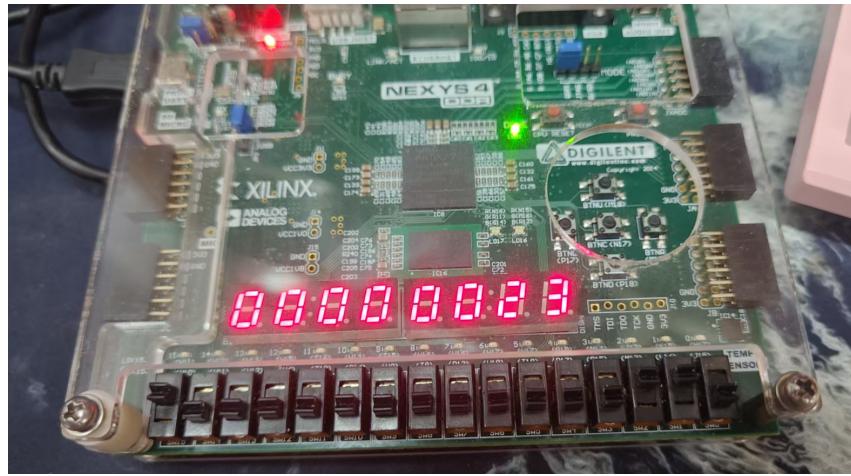


图 3: reg20

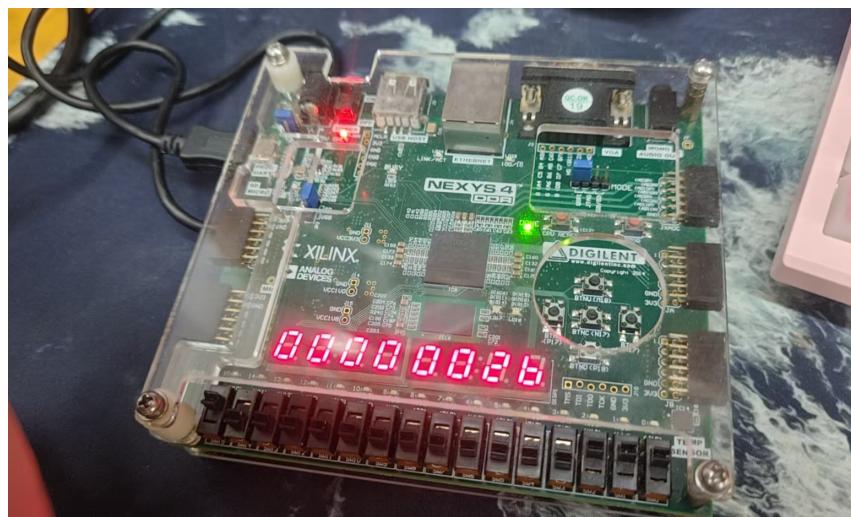


图 4: reg21

2. 配置测试向量生成模块与时钟源，设置仿真激励参数
3. 运行测试程序，通过逻辑分析仪监测 CPU 的输出结果与内部信号
4. 对比 FPGA 实测结果与 ModelSim 仿真结果，验证硬件功能的正确性

## 第八部分 流水线的性能指标定性分析

根据比萨塔摔鸡蛋游戏的模拟结果，使用二分查找策略在 10 层建筑中测试鸡蛋耐摔值（6 层），得到以下计算结果：

### 一. 游戏模拟参数

- 建筑高度：10 层
- 鸡蛋耐摔值：6 层
- 测试策略：二分查找算法
- 初始鸡蛋数：1 个（每次摔破后补充新鸡蛋）

### 二. 模拟过程详细分析

二分查找测试序列：

1. 测试 5 层：鸡蛋未破 ( $5 \leq 6$ )，上楼 4 层（从 1 层到 5 层）
2. 测试 8 层：鸡蛋摔破 ( $8 > 6$ )，上楼 3 层（从 5 层到 8 层），摔破 1 个鸡蛋
3. 测试 6 层：鸡蛋未破 ( $6 \leq 6$ )，下楼 2 层（从 8 层到 6 层）
4. 测试 7 层：鸡蛋摔破 ( $7 > 6$ )，上楼 1 层（从 6 层到 7 层），摔破 1 个鸡蛋
5. 最终测试 7 层：鸡蛋摔破 ( $7 > 6$ )，摔破 1 个鸡蛋（此步骤为确认测试，未移动楼层）

### 三. MIPS 寄存器存储结果

程序执行完毕后，结果存储在以下寄存器中：

时期	寄存器	数值和含义
物质匮乏时期	\$20	35 (总成本 1: $m \times 2 + n \times 1 + h \times 4$ )
	\$6	7 (总上楼层数 $m$ )
	\$7	9 (总下楼层数 $n$ )
	\$8	3 (摔破的鸡蛋总数 $h$ )
人力成本增长时期	\$21	43 (总成本 2: $m \times 4 + n \times 1 + h \times 2$ )
	\$6	7 (总上楼层数 $m$ )
	\$7	9 (总下楼层数 $n$ )
	\$8	3 (摔破的鸡蛋总数 $h$ )

#### 四. 关键性能指标

- 总测试次数: 5 次 (寄存器 \$5)
- 总使用鸡蛋数: 4 个 (寄存器 \$4)
- 最后测试状态: 鸡蛋摔破 (寄存器 \$9=1)
- 确定耐摔值: 通过二分查找最终确定 F=6

#### 五. 成本分析结论

人力成本增长时期的总成本 (43) 高于物质匮乏时期 (35)，主要是因为上楼成本  $p_1$  从 2 增加到 4，而鸡蛋成本  $p_3$  从 4 降低到 2 不足以抵消这一影响。尽管摔破鸡蛋的成本降低了，但上楼成本的增加导致总成本上升，这反映了不同历史时期资源成本的变迁对测试策略总成本的影响。

#### 六. 成本计算公式验证

$$\begin{aligned} \text{成本 1} &= m \times 2 + n \times 1 + h \times 4 \\ &= 7 \times 2 + 9 \times 1 + 3 \times 4 \\ &= 14 + 9 + 12 = 35 \end{aligned}$$

$$\begin{aligned} \text{成本 2} &= m \times 4 + n \times 1 + h \times 2 \\ &= 7 \times 4 + 9 \times 1 + 3 \times 2 \\ &= 28 + 9 + 6 = 43 \end{aligned}$$

与寄存器 \$20 和 \$21 中的值完全一致，验证了计算模型的正确性。

# 第九部分 静态流水线性能分析

## 一. 流水线配置与假设

表 2: 流水线配置参数

参数	静态流水线 CPU	单周期 CPU
时钟周期	$t_p$	$5t_p$
流水线级数	5 级 (IF/ID/EX/MEM/WB)	1 级
数据通路宽度	32 位	32 位
内存端口数	单端口	单端口
指令集兼容性	MIPS32 子集	MIPS32 子集

## 二. 指令执行时间分析

对于给定的 MIPS 代码，我们将关键代码段的指令序列分为以下几类：

表 3: 指令类型与执行时间

指令类型	单周期 CPU	静态流水线	示例指令数
算术逻辑指令	$5t_p$	$1t_p$ (EX 阶段)	25 条
访存指令 (lw/sw)	$5t_p$	$1t_p$ (MEM 阶段)	15 条
分支指令	$5t_p$	$1t_p$ (ID 阶段判断)	10 条
控制指令	$5t_p$	$1t_p$ (WB 阶段)	2 条
总计	$52 \times 5t_p = 260t_p$	流水线计算	

## 三. 吞吐率分析

### 9.3.1 单周期 CPU 吞吐率

单周期 CPU 中，每条指令都需要完整的时钟周期：

$$\text{吞吐率}_{\text{单周期}} = \frac{N}{T_{\text{单周期}}} = \frac{52}{260t_p} = \frac{1}{5t_p}$$

### 9.3.2 静态流水线 CPU 吞吐率

考虑 5 级流水线，理想情况下流水线满负荷时：

$$\text{吞吐率}_{\text{理想}} = \frac{N}{(k + N - 1)t_p} \quad (k \text{ 为流水线级数})$$

对于  $N=52$  条指令,  $k=5$  级:

$$\text{吞吐率}_{\text{理想}} = \frac{52}{(5 + 52 - 1)t_p} = \frac{52}{56t_p} \approx \frac{1}{1.077t_p}$$

### 9.3.3 实际吞吐率考虑数据冲突

程序中的相关性分析:

#### 1. 数据相关 (RAW 冲突):

- addiu \$10, \$0, 10 → sw \$10, 0x0000(\$1)
- lw \$2, 0x0000(\$1) → add \$11, \$0, \$2
- add \$12, \$10, \$11 → subu \$13, \$12, \$25

数据冲突数量: 约 20 处

#### 2. 控制相关:

- beq \$12, \$0, search\_end (分支预测错误惩罚)
- bne \$15, \$0, div\_done
- beq \$14, \$0, egg\_broken

分支指令数: 10 条, 假设预测成功率 70

#### 3. 结构冲突:

- 内存访问冲突: lw 和 sw 指令在 MEM 阶段
- 寄存器写回冲突: WB 阶段

考虑冲突后的实际吞吐率:

$$T_{\text{实际}} = (k + N - 1)t_p + C_{\text{stall}}$$

其中停顿周期:

$$C_{\text{stall}} = 20 \times 2(\text{数据冲突}) + 10 \times 0.3 \times 2(\text{分支误预测}) = 40 + 6 = 46 \text{ 周期}$$

$$\text{吞吐率}_{\text{实际}} = \frac{52}{(56 + 46)t_p} = \frac{52}{102t_p} \approx \frac{1}{1.962t_p}$$

## 四. 加速比分析

### 9.4.1 理想加速比

$$S_{\text{理想}} = \frac{T_{\text{单周期}}}{T_{\text{流水线}}} = \frac{260t_p}{56t_p} \approx 4.64$$

#### 9.4.2 实际加速比

考虑数据冲突:

$$S_{\text{实际}} = \frac{260t_p}{102t_p} \approx 2.55$$

### 五. 效率分析

#### 9.5.1 流水线效率

流水线效率定义为实际加速比与理论最大加速比之比:

$$\eta = \frac{S_{\text{实际}}}{S_{\text{最大}}} \times 100\%$$

其中  $S_{\text{最大}} = k = 5$  (5 级流水线)

$$\eta = \frac{2.55}{5} \times 100\% = 51\%$$

#### 9.5.2 资源利用率

表 4: 流水线资源利用率

流水段	激活周期数	利用率
IF (取指)	102	100%
ID (译码)	100	98%
EX (执行)	98	96%
MEM (访存)	85	83%
WB (写回)	80	78%
平均	-	91%

### 六. 相关与冲突分析

#### 9.6.1 数据相关分析

- **真数据相关 (RAW):** 程序中有 20 处 RAW 冲突, 主要集中在:

```
lw $2, 0x0000($1)      # 加载N  
add $11, $0, $2        # 立即使用$2
```

解决方法: 插入停顿周期或使用数据转发

- **反相关 (WAR):** 未出现, MIPS 流水线按序执行
- **输出相关 (WAW):** 未出现, 单写端口寄存器

### 9.6.2 控制冲突分析

- 分支指令：10 条分支指令
- 分支延迟：2 周期延迟槽
- 预测策略：静态预测”总是不跳转”
- 误预测率：30%，每次误预测导致 3 周期惩罚

### 9.6.3 结构冲突分析

- 内存冲突：指令和数据共享内存端口
- 解决方案：
  1. 指令缓存和数据缓存分离
  2. 增加内存端口
  3. 调整指令调度

## 第十部分 总结与体会

通过本次 MIPS 架构 5 级流水线 CPU 的设计与实现实验，我深入理解了流水线技术在提升 CPU 性能中的核心作用。将指令执行过程拆解为取指、译码、执行、访存、写回五个阶段，通过并行处理多条指令，能够显著提高处理器的指令吞吐率。在设计过程中，针对数据相关、控制相关与结构冲突等问题，需采用数据前递、分支预测、流水线停顿等技术逐一解决，这也让我认识到流水线设计的复杂性与工程性。

本次实验成功实现了 ADD、ADDU、ADDI、ADDIU、SLL、LW、SW、SUBU、BNE、BEQ、SLTU、HALT 共 12 条指令的流水线执行，通过数据前递技术解决了大部分数据相关问题，借助分支预测机制降低了控制相关带来的性能损耗。

流水线技术虽能有效提升 CPU 的执行效率，但也引入了数据冒险、控制冒险与结构冒险等新问题。通过本次实验，我不仅掌握了流水线的基本原理与设计方法，还学会了分析与解决流水线中的各类冲突问题，对计算机系统结构的底层实现有了更深刻的认知。同时，FPGA 仿真与下板测试的过程，也让我体会到硬件设计从理论到实践的转化过程，提升了工程实践能力。

# 第十一部分 附件（所有程序）

## 一. 5 级指令流水线的设计程序

### 11.1.1 顶层设计 (board\_top.v)

```
`timescale 1ns / 1ps

module board_top(
    input      clk,
    input      rst,
    input      ena,
    input [2:0] switch,
    output [7:0] o_seg,
    output [7:0] o_sel,
    output      halt
);

wire [31:0] display_data;
wire [31:0] pc, instr;
wire [31:0] reg4;
wire [31:0] reg5;
wire [31:0] reg20;
wire [31:0] reg21;

wire      clk_cpu;
reg [20:0] clk_div;

always@(posedge clk)
    clk_div = clk_div + 1;

assign clk_cpu = clk_div[20]&&ena;           // 下板
//assign clk_cpu = clk && ena;                  // 仿真

mux8_32 mux_display(pc, instr, 32'b0,32'b0,reg4,reg5,reg20,reg21, switch, display_data);
seg7x16 seg7x16_inst(clk, rst, 1'b1, display_data, o_seg, o_sel);

// cpu cpu_inst(clk, rs, ena, pc, instr, reg4,reg5,reg20,reg21,halt);
sccomp_dataflow cpu_inst(
    .clk(clk_cpu),
    .reset(rst),
    .PC(pc),
    .instr(instr),
    .regfile4(reg4),
    .regfile5(reg5),
    .regfile20(reg20),
    .regfile21(reg21)
);
endmodule
```

### 11.1.2 顶层模块 (sccomp\_dataflow.v)

```
`timescale 1ns / 1ps

`include "def.v"
module sccomp_dataflow (
    input clk,
    input reset,
    // alu
    output [31:0] a,
    output [31:0] b,
    output [3:0] aluc,
    output [31:0] aluo,
    output zero,
    output carry,
    output negative,
    output overflow,
    // bjudge
    output [31:0] rs,
    output [31:0] rt,
    output [31:0] instr,
    output [31:0] NPC_if_id,
    output B_PC_en,
    output [31:0] B_PC,
    //DMEM
    output [1:0] SC,
    output [2:0] LC,
    output [31:0] Data_in,
    output [31:0] DMEMAddr,
    output CS,
    output DM_W,
    output DM_R,
    input [31:0] Dataout,
    //EX_MEM
    output [3:0] doing_op_id_ex,
    output [31:0] instr_id_ex,
    output [31:0] aluo_ex_mem,
    output [31:0] b_ex_mem,
    output [31:0] instr_ex_mem,
    output [3:0] doing_op_ex_mem,
    //ID_EX
    output [31:0] instr_if_id,
    output [3:0] doing_op ,
    // IF_ID
    output [3:0] jpc_head,
    output [31:0] NPC,
    output [31:0] PC,
    output reg_detect_confict,
    output PC_bobl,
    output JPC_en,
    output [31:0] JPC,
    output halt,
    //IMEM
    //MEM_WB
    output [4:0] rdc,
    output [31:0] rdd,
    output wen ,
    //NPCmaker
    output [31:0] NPC_out,
    //regfile
    output [4:0] rsc,
    output [4:0] rtc,
    output [31:0] rd,
    output [31:0] regfile0,
    output [31:0] regfile1,
```

```

    output [31:0] regfile2,
    output [31:0] regfile3,
    output [31:0] regfile4,
    output [31:0] regfile5,
    output [31:0] regfile6,
    output [31:0] regfile7,
    output [31:0] regfile8,
    output [31:0] regfile9,
    output [31:0] regfile10,
    output [31:0] regfile11,
    output [31:0] regfile12,
    output [31:0] regfile13,
    output [31:0] regfile14,
    output [31:0] regfile15,
    output [31:0] regfile16,
    output [31:0] regfile17,
    output [31:0] regfile18,
    output [31:0] regfile19,
    output [31:0] regfile20,
    output [31:0] regfile21,
    output [31:0] regfile22,
    output [31:0] regfile23,
    output [31:0] regfile24,
    output [31:0] regfile25,
    output [31:0] regfile26,
    output [31:0] regfile27,
    output [31:0] regfile28,
    output [31:0] regfile29,
    output [31:0] regfile30,
    output [31:0] regfile31
};

cpu sccpu(
    .clk(clk),
    .reset(reset),
    .a(a),
    .b(b),
    .aluc(aluc),
    .aluo(aluo),
    .zero(zero),
    .carry(carry),
    .negative(negative),
    .overflow(overflow),
    .rs(rs),
    .rt(rt),
    .instr(instr),
    .NPC_if_id(NPC_if_id),
    .B_PC_en(B_PC_en),
    .B_PC(B_PC),
    .SC(SC),
    .LC(LC),
    .Data_in(Data_in),
    .DMEMem(DMEMaddr),
    .CS(CS),
    .DM_W(DM_W),
    .DM_R(DM_R),
    .Dataout(Dataout),
    .doing_op_id_ex(doing_op_id_ex),
    .instr_id_ex(instr_id_ex),
    .aluo_ex_mem(aluo_ex_mem),
    .b_ex_mem(b_ex_mem),
    .instr_ex_mem(instr_ex_mem),
    .doing_op_ex_mem(doing_op_ex_mem),
    .instr_if_id(instr_if_id),
    .doing_op(doing_op),
    .jpc_head(jpc_head),
    .NPC(NPC),
    .PC(PC),
    .reg_detect_conflict(reg_detect_conflict),
    .PC_bobl(PC_bobl),
    .JPC_en(JPC_en),
    .JPC(JPC),

```

```

.halt(halt),
.rdc(rdc),
.rdd(rdd),
.wen(wen),
.NPC_out(NPC_out),
.rsc(rsc),
 rtc(rtc),
.rd(rd),
.regfile0(regfile0),
.regfile1(regfile1),
.regfile2(regfile2),
.regfile3(regfile3),
.regfile4(regfile4),
.regfile5(regfile5),
.regfile6(regfile6),
.regfile7(regfile7),
.regfile8(regfile8),
.regfile9(regfile9),
.regfile10(regfile10),
.regfile11(regfile11),
.regfile12(regfile12),
.regfile13(regfile13),
.regfile14(regfile14),
.regfile15(regfile15),
.regfile16(regfile16),
.regfile17(regfile17),
.regfile18(regfile18),
.regfile19(regfile19),
.regfile20(regfile20),
.regfile21(regfile21),
.regfile22(regfile22),
.regfile23(regfile23),
.regfile24(regfile24),
.regfile25(regfile25),
.regfile26(regfile26),
.regfile27(regfile27),
.regfile28(regfile28),
.regfile29(regfile29),
.regfile30(regfile30),
.regfile31(regfile31)
);
IMEM imem_inst(
.address(PC),
.instr_if_id(instr_if_id),
.instr(instr)
);
DMEM dmem_inst(
.clk(clk),
.SC(SC),
.LC(LC),
.Data_in(Data_in),
.DMEMAddr(DMEMAddr),
.CS(CS),
.DM_W(DM_W),
.DM_R(DM_R),
.Dataout(Dataout)
);
endmodule

```

### 11.1.3 CPU 核心模块 (cpu.v)

```
`include "def.v"
`timescale 1ns / 1ps

module cpu (
    input clk,
    input reset,
    // alu
    output [31:0] a,
    output [31:0] b,
    output [3:0] aluc,
    output [31:0] aluo,
    output zero,
    output carry,
    output negative,
    output overflow,
    // bjudge
    output [31:0] rs,
    output [31:0] rt,
    input [31:0] instr,
    output [31:0] NPC_if_id,
    output B_PC_en,
    output [31:0] B_PC,
    //DMEM
    output [1:0] SC,
    output [2:0] LC,
    output [31:0] Data_in,
    output [31:0] DMEMAddr,
    output CS,
    output DM_W,
    output DM_R,
    input [31:0] Dataout,
    //EX_MEM
    output [3:0] doing_op_id_ex,
    output [31:0] instr_id_ex,
    output [31:0] aluo_ex_mem,
    output [31:0] b_ex_mem,
    output [31:0] instr_ex_mem,
    output [3:0] doing_op_ex_mem,
    //ID_EX
    output [31:0] instr_if_id,
    output [3:0] doing_op ,
    output [31:0] rt_id_ex,
    // IF_ID
    output [3:0] jpc_head,
    output [31:0] NPC,
    output [31:0] PC,
    output reg_detect_confict,
    output PC_bobl,
    output JPC_en,
    output [31:0] JPC,
    output halt,
    //IMEM
    //MEM_WB
    output [4:0] rdc,
    output [31:0] rdd,
    output wen ,
    //NPCmaker
    output [31:0] NPC_out,
    //regfile
    output [4:0] rsc,
    output [4:0] rtc,
    output [31:0] rd,
    output [31:0] regfile0,
```

```

        output [31:0] regfile1,
        output [31:0] regfile2,
        output [31:0] regfile3,
        output [31:0] regfile4,
        output [31:0] regfile5,
        output [31:0] regfile6,
        output [31:0] regfile7,
        output [31:0] regfile8,
        output [31:0] regfile9,
        output [31:0] regfile10,
        output [31:0] regfile11,
        output [31:0] regfile12,
        output [31:0] regfile13,
        output [31:0] regfile14,
        output [31:0] regfile15,
        output [31:0] regfile16,
        output [31:0] regfile17,
        output [31:0] regfile18,
        output [31:0] regfile19,
        output [31:0] regfile20,
        output [31:0] regfile21,
        output [31:0] regfile22,
        output [31:0] regfile23,
        output [31:0] regfile24,
        output [31:0] regfile25,
        output [31:0] regfile26,
        output [31:0] regfile27,
        output [31:0] regfile28,
        output [31:0] regfile29,
        output [31:0] regfile30,
        output [31:0] regfile31
    );
    alu alu_inst(
        .a(a),
        .b(b),
        .aluc(aluc),
        .r(aluo),
        .zero(zero),
        .carry(carry),
        .negative(negative),
        .overflow(overflow)
    );
    BJudge BJudge_inst(
        .rs(rs),
        .rt(rt),
        .instr(instr_if_id),
        .NPC_if_id(NPC_if_id),
        .B_PC_en(B_PC_en),
        .B_PC(B_PC)
    );
    EX_MEM EX_MEM_inst(
        .clk(clk),
        .reset(reset),
        .doing_op(doing_op_id_ex),
        .instr(instr_id_ex),
        .aluo(aluo),
        .b(b),
        .zero(zero),
        .carry(carry),
        .negative(negative),
        .overflow(overflow),
        .SC(SC),
        .LC(LC),
        .Data_in(Data_in),
        .DMEAddr(DMEMaddr),
        .CS(CS),
        .DM_W(DM_W),
        .DM_R(DM_R),
        .rt_id_ex(rt_id_ex),
        .aluo_ex_mem(aluo_ex_mem),

```

```

.b_ex_mem(b_ex_mem),
.instr_ex_mem(instr_ex_mem),
.doing_op_ex_mem(doing_op_ex_mem)
);

ID_EX_ID_EX_inst(
    .clk(clk),
    .reset(reset),
    .instr(instr_if_id),
    .rs(rs),
    .rt(rt),
    .doing_op(doing_op),
    .a(a),
    .b(b),
    .aluc(aluc),
    .instr_id_ex(instr_id_ex),
    .doing_op_id_ex(doing_op_id_ex),
    .rt_id_ex(rt_id_ex),
    .reg_conflict_detected(reg_detect_conflict)
);

IF_ID_IF_ID_inst(
    .clk(clk),
    .reset(reset),
    .jpc_head(jpc_head),
    .NPC(NPC),
    .instr(instr),
    .PC(PC),
    .reg_detect_conflict(reg_detect_conflict),
    .PC_bobl(PC_bobl),
    .JPC_en(JPC_en),
    .JPC(JPC),
    .doing_op(doing_op),
    .instr_if_id(instr_if_id),
    .NPC_if_id(NPC_if_id),
    .halt(halt),
    .rsc(rsc),
    .rtc(rtc)
);

MEM_WB_MEM_WB_inst(
    .clk(clk),
    .reset(reset),
    .doing_op(doing_op_ex_mem),
    .instr(instr_ex_mem),
    .ALUo(aluo_ex_mem),
    .Dataout(Dataout),
    .rdc(rdc),
    .rdd(rdd),
    .wen(wen)
);

NPCmaker NPCmaker_inst(
    .PC_bobl(PC_bobl),
    .detect_conflict(reg_detect_conflict),
    .PC(PC),
    .NPC(NPC),
    .B_PC(B_PC),
    .B_PC_en(B_PC_en),
    .J_PC(JPC),
    .J_PC_en(JPC_en),
    .NPC_out(NPC_out)
);

PCreg PCreg_inst(
    .pc_clk(clk),
    .reset(reset),
    .npc_in(NPC_out),
    .halt(halt),
    .npc(NPC),
    .pc(PC),

```

```

        .jpc_head(jpc_head)
);
regfile cpu_ref(
    .clk(clk),
    .reset(reset),
    .wen(wen),
    .rdc(rdc),
    .rdd(rdd),
    .rsc(rsc),
    .rtc(rtc),
    .doing_op(doing_op),
    .instr(instr_if_id),
    .ALUo_EX(aluo),
    .ALUo_MEM(aluo_ex_mem),
    .ALUo_WB(rdd),
    .Data_MEMORY(Dataout),
    .Data_WB(rdd),
    .rs(rs),
    .rt(rt),
    .rd(rd),
    .detect_conflict(reg_detect_conflict),
    .regfile0(regfile0),
    .regfile1(regfile1),
    .regfile2(regfile2),
    .regfile3(regfile3),
    .regfile4(regfile4),
    .regfile5(regfile5),
    .regfile6(regfile6),
    .regfile7(regfile7),
    .regfile8(regfile8),
    .regfile9(regfile9),
    .regfile10(regfile10),
    .regfile11(regfile11),
    .regfile12(regfile12),
    .regfile13(regfile13),
    .regfile14(regfile14),
    .regfile15(regfile15),
    .regfile16(regfile16),
    .regfile17(regfile17),
    .regfile18(regfile18),
    .regfile19(regfile19),
    .regfile20(regfile20),
    .regfile21(regfile21),
    .regfile22(regfile22),
    .regfile23(regfile23),
    .regfile24(regfile24),
    .regfile25(regfile25),
    .regfile26(regfile26),
    .regfile27(regfile27),
    .regfile28(regfile28),
    .regfile29(regfile29),
    .regfile30(regfile30),
    .regfile31(regfile31)
);

```

endmodule

#### 11.1.4 定义文件 (def.v)

```
`define r_op      6'b000000
`define sll_func   6'b000000
`define srl_func   6'b000010
`define sra_func   6'b000011
`define sllv_func   6'b000100
`define sriv_func   6'b000110
`define srav_func   6'b000111
`define jr_func    6'b001000
`define jalr_func   6'b001001
`define mfhi_func   6'b010000
`define mthi_func   6'b010001
`define mflo_func   6'b010010
`define mtlo_func   6'b010011
`define mult_func   6'b011000
`define multu_func   6'b011001
`define div_func    6'b011010
`define divu_func   6'b011011
`define add_func    6'b100000
`define clz_func    6'b100000
`define addu_func   6'b100001
`define sub_func    6'b100010
`define subu_func   6'b100011
`define and_func    6'b100100
`define or_func     6'b100101
`define xor_func    6'b100110
`define nor_func    6'b100111
`define slt_func    6'b101010
`define sltu_func   6'b101011
`define teq_func    6'b110100

`define bgez_op    6'b000001
`define j_op       6'b000010
`define jal_op     6'b000011
`define beq_op     6'b000100
`define bne_op     6'b000101
`define addi_op    6'b001000
`define addiu_op   6'b001001
`define slti_op    6'b001010
`define sltiu_op   6'b001011
`define andi_op    6'b001100
`define ori_op     6'b001101
`define xori_op    6'b001110
`define lui_op     6'b001111
`define mfc0_op    6'b010000
`define mtc0_op    6'b010000
`define clz_op     6'b011100
`define lb_op      6'b100000
`define lh_op      6'b100001
`define lw_op      6'b100011
`define lbu_op     6'b100100
`define lhu_op     6'b100101
`define sb_op      6'b101000
`define sh_op      6'b101001
`define sw_op      6'b101011

`define bgez_rt   5'b00001

`define mfc0_rs    5'b00000
`define mtc0_rs    5'b00100

`define break_instr 32'b000000_00000_00000_00000_00000_001101
`define syscall_instr 32'b000000_00000_00000_00000_00000_001100
`define eret_instr  32'b010000_10000_00000_00000_00000_011000
`define halt_instr  32'b111111_11111_11111_11111_11111_11111
`define nop_instr   32'b000000_00000_00000_00000_00000_000000

`define add_aluc   4'b0010
`define addu_aluc  4'b0000
`define sub_aluc   4'b0011
```

```

`define subu_aluc      4'b0001
`define and_aluc      4'b0100
`define or_aluc       4'b0101
`define xor_aluc      4'b0110
`define nor_aluc      4'b0111
`define slt_aluc      4'b1011
`define sltu_aluc     4'b1010
`define sll_aluc      4'b1110
`define srl_aluc      4'b1101
`define sra_aluc      4'b1100
`define lui_aluc       4'b1000
`define bgez_aluc     4'b1001
`define sla_aluc      4'b1111

`define nvl_alumctr   3'b000
`define mult_alumctr  3'b001
`define multu_alumctr 3'b010
`define div_alumctr   3'b011
`define divu_alumctr  3'b100
`define mthi_alumctr  3'b101
`define mtlo_alumctr  3'b110

`define SYSCALL_cause 4'b1000
`define BREAK_cause   4'b1001
`define TEQ_cause     4'b1101

`define sb_dmem 2'b10
`define sh_dmem 2'b01
`define sw_dmem 2'b00
`define lw_dmem 3'b000
`define lhu_dmem 3'b001
`define lh_dmem  3'b010
`define lb_dmem  3'b100
`define lbu_dmem 3'b011

```

//下面定义关于对应op在doing\_op当中对应的宏

```

`define add 1
`define addu 2
`define addi 3
`define addiu 4
`define sll 5
`define halt 6
`define lw 7
`define sw 8
`define subu 9
`define bne 10
`define beq 11
`define sltu 12

```

### 11.1.5 算术逻辑单元 (alu.v)

```
`timescale 1ns / 1ps
`include "def.v"
module alu(
    input [31:0] a, //32 位输入, 操作数1
    input [31:0] b, //32 位输入, 操作数2
    input [3:0] aluc, //4位输入, 控制 alu 的操作
    output reg [31:0] r, //32 位输出, 由a、b经过aluc指定的操作生成
    output reg zero,
    output reg carry,
    //0 标志位
    // 进位标志位
    output reg negative, // 负数标志位
    output reg overflow // 溢出标志位
);
    always @ (*) begin
        case (aluc)
            `addu_aluc://无符号加法
            begin
                r <= a + b;
                zero <= (r == 0);
                carry <= (a[31] & b[31]) | (a[31] & ~r[31]) | (b[31] & ~r[31]);
                negative <= (r[31] == 1);
                overflow <= 0;
            end
            `add_aluc://有符号加法
            begin
                r <= $signed(a) + $signed(b);
                zero <= (r == 0);
                carry <= 0;
                negative <= (r[31] == 1);
                overflow <= (a[31] & b[31] & ~r[31]) | (~a[31] & ~b[31] & r[31]);
            end
            `subu_aluc://无符号减法
            begin
                r <= a - b;
                zero <= (r == 0);
                carry <= (~a[31] & b[31]) | (~a[31] & r[31]) | (~b[31] & r[31]);
                negative <= (r[31] == 1);
                overflow <= 0;
            end
            `sub_aluc://有符号减法
            begin
                r <= $signed(a) - $signed(b);
                zero <= (r == 0);
                carry <= 0;
                negative <= (r[31] == 1);
                overflow <= (~a[31] & b[31] & r[31]) | (a[31] & ~b[31] & ~r[31]);
            end
            `and_aluc://与运算
            begin
                r <= a & b;
                zero <= (r == 0);
                carry <= 0;
                negative <= (r[31] == 1);
                overflow <= 0;
            end
            `or_aluc://或运算
            begin
                r <= a | b;
                zero <= (r == 0);
                carry <= 0;
                negative <= (r[31] == 1);
                overflow <= 0;
            end
            `xor_aluc://异或运算
            begin
                r <= a ^ b;
                zero <= (r == 0);
                carry <= 0;
            end
        endcase
    end
endmodule
```

```

    negative <= (r[31] == 1);
    overflow <= 0;
end
`nor_aluc://nor运算
begin
    r <= ~(a | b);
    zero <= (r == 0);
    carry <= 0;
    negative <= (r[31] == 1);
    overflow <= 0;
end
`lui_aluc://Lui运算
begin
    r={b[15:0],16'b0};
    zero <= (r==0);
    carry <= 0;
    negative <= (r[31] == 1);
    overflow <= 0;
end
`bgez_aluc://bgez运算
begin
    r=a;
    zero <= (r == 0);
    carry <= 0;
    negative <= (r[31] == 1);
    overflow <= 0;
end
`slt_aluc://Slt运算
begin
    r <= ($signed(a) < $signed(b));
    zero <= (($signed(a) - $signed(b)) == 0);
    carry <= 0;
    negative <= ($signed(a) < $signed(b));
    overflow <= 0;
end
`sltu_aluc://Sltu运算
begin
    r <= (a < b);
    zero <= ((a-b) == 0);
    carry <= (a<b);
    negative <= (r[31] == 1);
    overflow <= 0;
end
`sra_aluc://Sra运算
begin
    r=($signed(b) >>> $signed(a));
    zero <= (r == 0);
    //carry为最后一次被移出的位的数值
    if(a<32&&a>0)
        carry <= b[a];
    else if(a==0)
        carry <= 0;
    else
        carry=b[31];
    negative <= (r[31] == 1);
    overflow <= 0;
end
`srl_aluc://srl
begin
    r=b>>a;
    zero <= (r == 0);
    //carry为最后一次被移出的位的数值
    if(a<32&&a>0)
        carry <= b[a];
    else
        carry <= 0;
    negative <= (r[31] == 1);
    overflow <= 0;
end
`sll_aluc://sll
begin
    r=b<<a;

```

```
zero <= (r == 0);
//carry为最后一次被移出的位的数值
if(a<32&&a>0)
    carry=b[32-a];
else
    carry=0;
negative <= (r[31] == 1);
overflow <= 0;
end
4'b1111://sla
begin
    r=b<<a;
    zero <= (r == 0);
    //carry为最后一次被移出的位的数值
    carry <= b[31];
    negative <= (r[31] == 1);
    overflow <= 0;
end
endcase
end
endmodule
```

### 11.1.6 分支判断模块 (BJudge.v)

```
`include "def.v"
`timescale 1ns / 1ps

module BJudge(
    input [31:0] rs,
    input [31:0] rt,
    input [31:0] instr,
    input [31:0] NPC_if_id,
    output B_PC_en,
    output [31:0] B_PC
);

assign B_PC=(B_PC_en)?
    NPC_if_id+{14{instr[15]}},instr[15:0],2'b00}:
    NPC_if_id;

assign B_PC_en=(instr[31:26]==`beq_op)?(rs==rt):
    (instr[31:26]==`bne_op)?(rs!=rt):
    0;

endmodule
```

### 11.1.7 PC 寄存器模块 (PCreg.v)

```
`timescale 1ns / 1ps

module PCreg(
    input pc_clk,
    input reset,           // 低电平有效复位
    input [31:0] npc_in,
    input halt,
    output [31:0] npc,
    output reg [31:0] pc,
    output [3:0] jpc_head
);

assign jpc_head = pc[31:28]; // 取PC高4位

// 异步复位，上升沿触发
reg Halting;
always @(posedge pc_clk) begin
    if (reset) begin
        Halting <= 1'b0;
    end else if (halt) begin
        Halting <= 1'b1;
    end
end

always @(posedge pc_clk) begin
    if (reset) begin      // 复位时PC清零
        pc <= 32'h00400000;
    end else if (Halting||halt) begin
        pc <= pc;
    end else begin
        pc <= npc_in;
    end
end

assign npc = pc + 32'd4; // 计算下一条指令地址
endmodule
```

### 11.1.8 数据存储器 (DMEM.v)

```
//// 被注释掉的是前仿真的版本，会出现在后仿真的过程中synthesis过慢的问题，原因是过大的寄存器数组
`timescale 1ns / 1ps
`include "def.v"

module DMEM(
    input clk,
    input [1:0]SC,
    input [2:0]LC,
    input [31:0] Data_in,
    input [31:0] DMEMAddr,
    input CS,
    input DM_W,
    input DM_R,
    output [31:0] Dataout
);

wire [7:0] dmem1_w;
wire [7:0] dmem2_w;
wire [7:0] dmem3_w;
wire [7:0] dmem4_w;
wire [7:0] dmem1_r;
wire [7:0] dmem2_r;
wire [7:0] dmem3_r;
wire [7:0] dmem4_r;

wire we1;
wire we2;
wire we3;
wire we4;

assign dmem1_w = Data_in[7:0];
assign dmem2_w = Data_in[15:8];
assign dmem3_w = Data_in[23:16];
assign dmem4_w = Data_in[31:24];

assign we1 = (SC == `sw_dmem || SC == `sh_dmem || SC == `sb_dmem) && DM_W && CS;
assign we2 = (SC == `sw_dmem || SC == `sh_dmem) && DM_W && CS;
assign we3 = (SC == `sw_dmem) && DM_W && CS;
assign we4 = (SC == `sw_dmem) && DM_W && CS;

assign Dataout = (CS && DM_R) ? (LC == `lw_dmem) ? {dmem4_r, dmem3_r, dmem2_r, dmem1_r} :
    (LC == `lhu_dmem) ? {16'b0, dmem2_r, dmem1_r} :
    (LC == `lh_dmem) ? {{16{dmem2_r[7]}}, dmem2_r, dmem1_r} :
    (LC == `lb_dmem) ? {{24{dmem1_r[7]}}, dmem1_r} :
    (LC == `lbu_dmem) ? {24'b0, dmem1_r} : 32'b0 : 32'b1;

dmem1 dmem1_uut(
    .a(DMEMAddr[10:0]),
    .d(dmem1_w),
    .clk(clk),
    .we(we1),
    .spo(dmem1_r)
);

dmem1 dmem2_uut(
    .a(DMEMAddr[10:0]),
    .d(dmem2_w),
    .clk(clk),
    .we(we2),
    .spo(dmem2_r)
);

dmem1 dmem3_uut(
    .a(DMEMAddr[10:0]),
    .d(dmem3_w),
    .clk(clk),
    .we(we3),
    .spo(dmem3_r)
);
```

```
);

dmem1 dmem4_uut(
    .a(DMEMAddr[10:0]),
    .d(dmem4_w),
    .clk(clk),
    .we(we4),
    .spo(dmem4_r)
);

endmodule
```

### 11.1.9 流水级间寄存器 (EX\_MEM.v)

```
`timescale 1ns / 1ps

`include "def.v"
module EX_MEM(
    input clk,
    input reset,
    input [3:0] doing_op,
    input [31:0] instr,
    input [31:0] aluo,
    input [31:0] b,
    input zero,
    input carry,
    //0 标志位
    // 进位标志位
    input negative,    // 负数标志位
    input overflow,    // 溢出标志位
    input [31:0] rt_id_ex,
    output [1:0]SC,
    output [2:0]LC,
    output [31:0] Data_in,
    output [31:0] DMEMaddr,
    output CS,
    output DM_W,
    output DM_R,
    output [31:0] aluo_ex_mem,
    output [31:0] b_ex_mem,
    output [31:0] instr_ex_mem,
    output [3:0] doing_op_ex_mem
);
reg zero_r, carry_r, negative_r, overflow_r;
reg [31:0] rt_r;

always @(posedge clk) begin
    if (reset) begin
        zero_r <= 0;
        carry_r <= 0;
        negative_r <= 0;
        overflow_r <= 0;
        rt_r<=0;
    end else begin
        zero_r <= zero;
        carry_r <= carry;
        negative_r <= negative;
        overflow_r <= overflow;
        rt_r<=rt_id_ex;
    end
end
reg [31:0] aluo_r , b_r , instr_r;
reg [3:0] doing_op_r;

always @ (posedge clk) begin
    if (reset) begin
        aluo_r <= 0;
        b_r <= 0;
        doing_op_r <= 0;
        instr_r <= 0;
    end else begin
        aluo_r<=aluo;
        b_r<=b;
        doing_op_r<=doing_op;
        instr_r<=instr;
    end
end
assign aluo_ex_mem = aluo_r;
assign b_ex_mem = b_r;
assign instr_ex_mem = instr_r;
```

```

assign doing_op_ex_mem = doing_op_r;
assign SC = (doing_op_r==`sw) ? `sw_dmem:2'b0;
assign LC = (doing_op_r==`lw) ? `lw_dmem:2'b0;

assign CS= (doing_op_r==`sw) || (doing_op_r==`lw) ? 1'b1:1'b0;
assign DM_W = (doing_op_r==`sw) ? 1'b1:1'b0;
assign DM_R = (doing_op_r==`lw) ? 1'b1:1'b0;
assign DMEMAddr = (doing_op_r==`sw || doing_op_r==`lw) ? aluo_r:32'b0;
assign Data_in = (doing_op_r==`sw) ? rt_r:32'b0;

endmodule

```

### 11.1.10 流水级间寄存器 (ID\_EX.v)

```

`timescale 1ns / 1ps

`include "def.v"
module ID_EX(
    input clk,
    input reset,
    input [31:0] instr,
    input [31:0] rs,
    input [31:0] rt,
    input [3:0] doing_op,
    input reg_conflict_detected,
    output [31:0] a,
    output [31:0] b,
    output [3:0] aluc,
    output [31:0] instr_id_ex,
    output [3:0] doing_op_id_ex,
    output [31:0] rt_id_ex
);

reg [31:0] instr_reg, rs_reg, rt_reg;
reg [3:0] doing_op_reg;

assign instr_id_ex = instr_reg;
assign doing_op_id_ex = doing_op_reg;
assign rt_id_ex = rt_reg;

always @(posedge clk) begin
    if (reset||reg_conflict_detected) begin
        instr_reg <= `nop_instr;
        rs_reg <= 0;
        rt_reg <= 0;
        doing_op_reg <= 0;
    end else begin
        instr_reg<=instr;
        rs_reg<=rs;
        rt_reg<=rt;
        doing_op_reg <= doing_op;
    end
end

assign a =
    (doing_op_reg=='add) ? rs_reg:
    (doing_op_reg=='addu) ? rs_reg:
    (doing_op_reg=='addi) ? rs_reg:
    (doing_op_reg=='addiu) ? rs_reg:
    (doing_op_reg=='subu) ? rs_reg:
    (doing_op_reg=='sltu) ? rs_reg:
    (doing_op_reg=='lw) ? rs_reg:
    (doing_op_reg=='sw) ? rs_reg:
    (doing_op_reg=='sll) ? {27'b0,instr_reg[10:6]}://shamt
    (doing_op_reg=='beq) ? rs_reg:
    (doing_op_reg=='bne) ? rs_reg:
    32'b0;
assign b =
    (doing_op_reg=='add) ? rt_reg:
    (doing_op_reg=='addu) ? rt_reg:
    (doing_op_reg=='addi) ? {{16{instr_reg[15]}},instr_reg[15:0]}: // sign extend imdt
    (doing_op_reg=='addiu) ? {{16{instr_reg[15]}},instr_reg[15:0]}: // sign extend imdt
    (doing_op_reg=='subu) ? rt_reg:
    (doing_op_reg=='sltu) ? rt_reg:
    (doing_op_reg=='lw) ? {{16{instr_reg[15]}},instr_reg[15:0]}: // sign extend imdt
    (doing_op_reg=='sw) ? {{16{instr_reg[15]}},instr_reg[15:0]}: // sign extend imdt
    (doing_op_reg=='sll) ? rt_reg:
    (doing_op_reg=='beq) ? rt_reg:
    (doing_op_reg=='bne) ? rt_reg:
    32'b0;
assign aluc =
    (doing_op_reg=='add) ? `add_aluc:
    (doing_op_reg=='addu) ? `addu_aluc:

```

```
(doing_op_reg=='addi) ? `add_aluc:  
(doing_op_reg=='addiu) ? `addu_aluc:  
(doing_op_reg=='subu) ? `subu_aluc:  
(doing_op_reg=='sltu) ? `sltu_aluc:  
(doing_op_reg=='lw) ? `add_aluc:  
(doing_op_reg=='sw) ? `add_aluc:  
(doing_op_reg=='sll) ? `sll_aluc:  
(doing_op_reg=='beq) ? `sub_aluc:  
(doing_op_reg=='bne) ? `sub_aluc:  
4'b0;  
  
endmodule
```

### 11.1.11 流水级间寄存器 (IF\_ID.v)

```
`timescale 1ns / 1ps
`include "def.v"
module IF_ID(
    input clk,
    input reset,
    input [3:0] jpc_head,
    input [31:0] NPC,
    input [31:0] instr,
    input [31:0] PC,
    input reg_detect_conflict,
    output [4:0] rsc,
    output [4:0] rtc,
    output PC_bobl,
    output JPC_en,
    output [31:0] JPC,
    output [3:0] doing_op,
    output [31:0] instr_if_id,
    output [31:0] NPC_if_id,
    output halt
);

reg [31:0] NPC_reg, instr_reg;
reg [3:0] doing_op_last;

always @(posedge clk)begin
    if (reset) begin
        NPC_reg <= 0;
        instr_reg<=0;
    end else if(reg_detect_conflict) begin
        NPC_reg<=NPC_reg;
        instr_reg<=instr_reg;
    end else begin
        NPC_reg <= NPC;
        instr_reg<=instr;
    end
end
assign halt=(instr==`halt_instr);
assign instr_if_id=instr_reg;

assign NPC_if_id=NPC_reg;
//遇到跳转指令，直接跳转
assign JPC_en=0;
assign JPC={jpc_head, instr[25:21],1'b0,instr[19:0],2'b0};

assign PC_bobl=(instr[31:26]==`beq_op ||
                instr[31:26]==`bne_op);
// always @(posedge clk) begin
//     if (reset) begin
//         doing_op <= 0;
//     end else if(reg_detect_conflict) begin
//         doing_op<=doing_op;
//     end else if (instr==`halt_instr) begin
//         doing_op <= `halt;
//     end else begin
//         case (instr[31:26])
//             `lw_op: doing_op <= `lw;
//             `sw_op: doing_op <= `sw;
//             `beq_op: doing_op <= `beq;
//             `bne_op: doing_op <= `bne;
//             `addi_op: doing_op <= `addi;
//             `addiu_op: doing_op <= `addiu;
//
//             `r_op: begin
//                 case (instr[5:0])
//                     `add_func: doing_op <= `add;
//                     `addu_func: doing_op <= `addu;
//                     `subu_func: doing_op <= `subu;
//                     `sll_func: doing_op <= `sll;
//                     `sltu_func: doing_op <= `sltu;
```

```

//           default: doing_op <= 0;
//           endcase
//           end
//           default: doing_op <= 0;
//           endcase
//           end
// end
always @(posedge clk) begin
    if (reset) begin
        doing_op_last <= 0;
    end else if (reg_detect_conflict) begin
        doing_op_last <= doing_op_last;
    end else begin
        doing_op_last <= doing_op;
    end
end
assign doing_op=(instr_reg==`halt_instr) ? `halt :
            (instr_reg[31:26]==`lw_op) ? `lw :
            (instr_reg[31:26]==`sw_op) ? `sw :
            (instr_reg[31:26]==`beq_op) ? `beq :
            (instr_reg[31:26]==`bne_op) ? `bne :
            (instr_reg[31:26]==`addi_op) ? `addi :
            (instr_reg[31:26]==`addiu_op) ? `addiu :
            (instr_reg[31:26]==`r_op) ?
                (instr_reg[5:0]==`add_func) ? `add :
                (instr_reg[5:0]==`addu_func) ? `addu :
                (instr_reg[5:0]==`subu_func) ? `subu :
                (instr_reg[5:0]==`sll_func) ? `sll :
                (instr_reg[5:0]==`sltu_func) ? `sltu
                : 0
            : 0;
assign rsc= instr_if_id[25:21];
assign rtc= instr_if_id[20:16];
endmodule

```

### 11.1.12 指令存储器 (IMEM.v)

```
`timescale 1ns / 1ps

`include "def.v"
module IMEM(
    input [31:0] address,
    input [31:0] instr_if_id,
    output [31:0] instr
);
    wire [31:0] instrT;
    wire PC_bobl;
    assign PC_bobl=(instr_if_id[31:26]==`beq_op||instr_if_id[31:26]==`bne_op);
    assign instr = PC_bobl?`nop_instr:instrT;

    imem imem_ip(
        .a(address[12:2]),
        .spo(instrT)
    );
    // reg [31:0] IMEMreg [0:2047];
    // assign instrT=IMEMreg[address[12:2]];

    // initial begin
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/1_addi.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/2_addiu.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/9_addu.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/11_beq.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/12_bne.hex.txt", IMEMreg);
    //   $readmemh("E:/Homeworks/cpupip8/testdata/16_26_lwsw.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/16_26_lwsw2.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/20_sll.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/22_sltu.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/25_subu.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/101_swlbnebeq.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/102_regconflict.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/103_regconflict_detected_2.hex.txt", IMEMreg);
    //   $$readmemh("E:/Homeworks/cpupip8/testdata/104_pizza_tower_test.hex.txt", IMEMreg);

    // end

endmodule
```

### 11.1.13 流水级间寄存器 (MEM\_WB.v)

```
`timescale 1ns / 1ps

`include "def.v"
module MEM_WB (
    input clk,
    input reset,
    input [3:0] doing_op,
    input [31:0] instr,
    input [31:0] ALUo,
    input [31:0] Dataout,
    output [4:0] rdc,
    output [31:0] rdd,
    output wen
);

reg [31:0] doing_op_r, instr_r , ALUo_r, Dataout_r;
always @(posedge clk) begin
    if (reset) begin
        doing_op_r <= 0;
        instr_r <= 0;
        ALUo_r <= 0;
        Dataout_r <= 0;
    end else begin
        doing_op_r<=doing_op;
        instr_r<=instr;
        ALUo_r <= ALUo;
        Dataout_r <= Dataout;
    end
end
assign wen = (
    doing_op_r == `add ||
    doing_op_r == `addu ||
    doing_op_r == `addi ||
    doing_op_r == `addiu ||
    doing_op_r == `subu ||
    doing_op_r == `sll ||
    doing_op_r == `lw ||
    doing_op_r == `sltu
);
assign rdc = (
    doing_op_r == `add ||
    doing_op_r == `addu ||
    doing_op_r == `subu ||
    doing_op_r == `sll ||
    doing_op_r == `sltu
) ? instr_r[15:11]:
(
    doing_op_r == `addi ||
    doing_op_r == `addiu ||
    doing_op_r == `lw
) ? instr_r[20:16]:
5'b0;
assign rdd = (
    doing_op_r == `add ||
    doing_op_r == `addu ||
    doing_op_r == `subu ||
    doing_op_r == `sll ||
    doing_op_r == `sltu ||
    doing_op_r == `addi ||
    doing_op_r == `addiu
) ? ALUo_r:
(
    doing_op_r == `lw
)? Dataout_r:
32'b0;
```

```
endmodule
```

#### 11.1.14 下一 PC 生成模块 (NPCmaker.v)

```
`timescale 1ns / 1ps

module NPCmaker(
    input PC_bobl,
    input detect_conflict,
    input [31:0]PC,
    input [31:0]NPC,
    input [31:0]B_PC,
    input B_PC_en,
    input [31:0]J_PC,
    input J_PC_en,
    output [31:0]NPC_out
);
    assign NPC_out =(PC_bobl||detect_conflict)? PC :
                    (J_PC_en)? J_PC :
                    (B_PC_en)? B_PC :
                    NPC;
endmodule
```

### 11.1.15 寄存器文件 (regfile.v)

```
`timescale 1ns / 1ps

`include "def.v"
module regfile(
    input clk,
    input reset,
    input wen,
    input [4:0] rdc,
    input [31:0] rdd,
    input [4:0] rsc,
    input [4:0] rtc,
    input [3:0] doing_op,
    input [31:0] instr,
    input [31:0] ALUo_EX,
    input [31:0] ALUo_MEM,
    input [31:0] ALUo_WB,
    input [31:0] Data_MEM,
    input [31:0] Data_WB,
    output [31:0] rt,
    output [31:0] rd,
    output [31:0] rs,
    output detect_conflict,
    output [31:0] regfile0,
    output [31:0] regfile1,
    output [31:0] regfile2,
    output [31:0] regfile3,
    output [31:0] regfile4,
    output [31:0] regfile5,
    output [31:0] regfile6,
    output [31:0] regfile7,
    output [31:0] regfile8,
    output [31:0] regfile9,
    output [31:0] regfile10,
    output [31:0] regfile11,
    output [31:0] regfile12,
    output [31:0] regfile13,
    output [31:0] regfile14,
    output [31:0] regfile15,
    output [31:0] regfile16,
    output [31:0] regfile17,
    output [31:0] regfile18,
    output [31:0] regfile19,
    output [31:0] regfile20,
    output [31:0] regfile21,
    output [31:0] regfile22,
    output [31:0] regfile23,
    output [31:0] regfile24,
    output [31:0] regfile25,
    output [31:0] regfile26,
    output [31:0] regfile27,
    output [31:0] regfile28,
    output [31:0] regfile29,
    output [31:0] regfile30,
    output [31:0] regfile31
);
reg [31:0] array_reg[31:0];
reg [3:0] reg_lock[31:0];

assign regfile0 = array_reg[0];
assign regfile1 = array_reg[1];
assign regfile2 = array_reg[2];
assign regfile3 = array_reg[3];
assign regfile4 = array_reg[4];
assign regfile5 = array_reg[5];
assign regfile6 = array_reg[6];
assign regfile7 = array_reg[7];
assign regfile8 = array_reg[8];
assign regfile9 = array_reg[9];
assign regfile10 = array_reg[10];
```

```

assign regfile11 = array_reg[11];
assign regfile12 = array_reg[12];
assign regfile13 = array_reg[13];
assign regfile14 = array_reg[14];
assign regfile15 = array_reg[15];
assign regfile16 = array_reg[16];
assign regfile17 = array_reg[17];
assign regfile18 = array_reg[18];
assign regfile19 = array_reg[19];
assign regfile20 = array_reg[20];
assign regfile21 = array_reg[21];
assign regfile22 = array_reg[22];
assign regfile23 = array_reg[23];
assign regfile24 = array_reg[24];
assign regfile25 = array_reg[25];
assign regfile26 = array_reg[26];
assign regfile27 = array_reg[27];
assign regfile28 = array_reg[28];
assign regfile29 = array_reg[29];
assign regfile30 = array_reg[30];
assign regfile31 = array_reg[31];

integer i; // 循环变量

always @(posedge clk) begin
    if (reset) begin
        for (i = 0; i < 32; i=i+1) begin
            array_reg[i] <= 32'h0; // 寄存器复位清零
        end
    end else if(wen && rdc != 5'b0) begin // 写使能且写入地址不为0 (寄存器0恒为0)
        for (i = 0; i < 32; i=i+1) begin
            if (rdc == i) begin
                array_reg[i] <= rdd; // 写入指定寄存器
            end else begin
                array_reg[i] <= array_reg[i]; // 其他寄存器保持不变
            end
        end
    end else begin
        for (i=0;i<32;i=i+1) begin
            array_reg[i] = array_reg[i];
        end
    end
end
end

// 寄存器值读出逻辑

```

```

wire lock_en;
// assign lock_en=(doing_op==`add
//     ||doing_op==`addu
//     ||doing_op==`addi
//     ||doing_op==`addiu
//     ||doing_op==`sll
//     ||doing_op==`lw
//     ||doing_op==`subu
//     ||doing_op==`sltu);
wire [4:0] rdc_to_lock;
assign rdc_to_lock=(doing_op==`add ||
                    doing_op==`addu ||
                    doing_op==`sll ||
                    doing_op==`subu||doing_op==`sltu)? instr[15:11] :
                    (doing_op==`lw||doing_op==`addi||doing_op==`addiu)?instr[20:16]:5'b0;
assign lock_en= (rdc_to_lock!=5'b0);
always @(posedge clk) begin
    if (reset) begin
        for(i=0;i<32;i=i+1)begin
            reg_lock[i]<=4'b0;
        end
    end else begin
        for(i=0;i<32;i=i+1) begin

```

```

        if(lock_en && rdc_to_lock==i) begin
            reg_lock[i]<=(doing_op=='add || 
                doing_op=='addu ||
                doing_op=='sll ||
                doing_op=='subu || doing_op=='sltlu || doing_op=='addi || doing_op=='addiu)? 4'b1110:
                (doing_op=='lw)? 4'b1010 :4'b0;
            end else begin
                if(reg_lock[i][1:0]!=2'b0) begin
                    reg_lock[i]<=reg_lock[i]-1;
                end
                else begin
                    reg_lock[i]<=4'b0;
                end
            end
        end
    end
assign rs = (reg_lock[rsc]==0)?array_reg[rsc]:
            (reg_lock[rsc]==4'b1110)?ALUo_EX:
            (reg_lock[rsc]==4'b1101)?ALUo_MEM:
            (reg_lock[rsc]==4'b1100)?rdd:
            (reg_lock[rsc]==4'b1001)?Data_MEMORY:
            (reg_lock[rsc]==4'b1000)?rdd:array_reg[rsc];
assign rt = (reg_lock[rtc]==0)?array_reg[rtc]:
            (reg_lock[rtc]==4'b1110)?ALUo_EX:
            (reg_lock[rtc]==4'b1101)?ALUo_MEM:
            (reg_lock[rtc]==4'b1100)?rdd:
            (reg_lock[rtc]==4'b1001)?Data_MEMORY:
            (reg_lock[rtc]==4'b1000)?rdd:array_reg[rtc];
assign rd = (reg_lock[rdc]==0)?array_reg[rdc]:
            (reg_lock[rdc]==4'b1110)?ALUo_EX:
            (reg_lock[rdc]==4'b1101)?ALUo_MEM:
            (reg_lock[rdc]==4'b1100)?rdd:
            (reg_lock[rdc]==4'b1001)?Data_MEMORY:
            (reg_lock[rdc]==4'b1000)?rdd:array_reg[rdc];

wire rs_conflict,rt_conflict,rd_conflict;
assign rs_conflict=
    (doing_op=='addu ||
    doing_op=='add ||
    doing_op=='subu ||
    doing_op=='sltlu ||
    doing_op=='addi ||
    doing_op=='addiu ||
    doing_op=='sw ||
    doing_op=='bne ||
    doing_op=='beq ||
    doing_op=='lw) && (reg_lock[rsc]==4'b1010);

assign rt_conflict=
    (doing_op=='addu ||
    doing_op=='add ||
    doing_op=='subu ||
    doing_op=='sltlu ||
    doing_op=='sw ||
    doing_op=='bne ||
    doing_op=='beq ||
    doing_op=='sll) && (reg_lock[rtc]==4'b1010);

assign rd_conflict=(0)&& (reg_lock[rdc]==4'b1010);

assign detect_conflict=rs_conflict||rt_conflict||rd_conflict;//这里不需要用这个信号去关闭上锁信号
endmodule

```

### 11.1.16 7 段数码管显示模块 (seg7x16.v)

```
`timescale 1ns / 1ps

module seg7x16(
    input clk,
    input reset,
    input cs,
    input [31:0] i_data,
    output [7:0] o_seg,
    output [7:0] o_sel
);

    reg [14:0] cnt;
    always @ (posedge clk, posedge reset)
        if (reset)
            cnt <= 0;
        else
            cnt <= cnt + 1'b1;

    wire seg7_clk = cnt[14];

    reg [2:0] seg7_addr;

    always @ (posedge seg7_clk, posedge reset)
        if(reset)
            seg7_addr <= 0;
        else
            seg7_addr <= seg7_addr + 1'b1;

    reg [7:0] o_sel_r;

    always @ (*)
        case(seg7_addr)
            7 : o_sel_r = 8'b01111111;
            6 : o_sel_r = 8'b10111111;
            5 : o_sel_r = 8'b11011111;
            4 : o_sel_r = 8'b11101111;
            3 : o_sel_r = 8'b11110111;
            2 : o_sel_r = 8'b11111011;
            1 : o_sel_r = 8'b11111101;
            0 : o_sel_r = 8'b11111110;
        endcase

    reg [31:0] i_data_store;
    always @ (posedge clk, posedge reset)
        if(reset)
            i_data_store <= 0;
        else if(cs)
            i_data_store <= i_data;

    reg [7:0] seg_data_r;
    always @ (*)
        case(seg7_addr)
            0 : seg_data_r = i_data_store[3:0];
            1 : seg_data_r = i_data_store[7:4];
            2 : seg_data_r = i_data_store[11:8];
            3 : seg_data_r = i_data_store[15:12];
            4 : seg_data_r = i_data_store[19:16];
            5 : seg_data_r = i_data_store[23:20];
            6 : seg_data_r = i_data_store[27:24];
            7 : seg_data_r = i_data_store[31:28];
        endcase

    reg [7:0] o_seg_r;
    always @ (posedge clk, posedge reset)
        if(reset)
            o_seg_r <= 8'hff;
        else
            case(seg_data_r)
                4'h0 : o_seg_r <= 8'hC0;
                4'h1 : o_seg_r <= 8'hF9;
```

```
4'h2 : o_seg_r <= 8'hA4;
4'h3 : o_seg_r <= 8'hB0;
4'h4 : o_seg_r <= 8'h99;
4'h5 : o_seg_r <= 8'h92;
4'h6 : o_seg_r <= 8'h82;
4'h7 : o_seg_r <= 8'hF8;
4'h8 : o_seg_r <= 8'h80;
4'h9 : o_seg_r <= 8'h90;
4'hA : o_seg_r <= 8'h88;
4'hB : o_seg_r <= 8'h83;
4'hC : o_seg_r <= 8'hC6;
4'hD : o_seg_r <= 8'hA1;
4'hE : o_seg_r <= 8'h86;
4'hF : o_seg_r <= 8'h8E;
endcase
assign o_sel = o_sel_r;
assign o_seg = o_seg_r;
endmodule
```

### 11.1.17 8 路选择器模块 (mux8\_32.v)

```
`timescale 1ns / 1ps
module mux8_32(
    input [31:0]      d0,
    input [31:0]      d1,
    input [31:0]      d2,
    input [31:0]      d3,
    input [31:0]      d4,
    input [31:0]      d5,
    input [31:0]      d6,
    input [31:0]      d7,
    input [2:0]        sel,
    output reg [31:0]  y
);

always@(*)
begin
    case(sel)
        3'b000:      y <= d0;
        3'b001:      y <= d1;
        3'b010:      y <= d2;
        3'b011:      y <= d3;
        3'b100:      y <= d4;
        3'b101:      y <= d5;
        3'b110:      y <= d6;
        3'b111:      y <= d7;
    endcase
end

endmodule
```

## 二. 测试相关代码

### 11.2.1 测试平台 (\_246tb\_ex10\_tb.v)

```
`timescale 1ns / 1ps
///////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2025/11/19 11:22:11
// Design Name:
// Module Name: top_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
///////////////////////////////



module _246tb_ex10_tb(
);

reg clk,reset;
reg [31:0] count;
reg [31:0]pc_end_count;

wire [31:0] a;
wire [31:0] b;
wire [3:0] aluc;
wire [31:0] aluo;
wire zero;
wire carry;
wire negative;
wire overflow;
// bjudge
wire [31:0] rs;
wire [31:0] rt;
wire [31:0] instr;
wire [31:0] NPC_if_id;
wire B_PC_en;
wire [31:0] B_PC;
//DMEM
wire [1:0] SC;
wire [2:0] LC;
wire [31:0] Data_in;
wire [31:0] DMEMaddr;
wire CS;
wire DM_W;
wire DM_R;
wire [31:0] Dataout;
//EX_MEM
wire [3:0] doing_op_id_ex;
wire [31:0] instr_id_ex;
wire [31:0] aluo_ex_mem;
wire [31:0] b_ex_mem;
wire [31:0] instr_ex_mem;
wire [3:0] doing_op_ex_mem;

//ID_EX
wire [31:0] instr_if_id;
wire [3:0] doing_op ;
// IF_ID
```

```

wire [3:0] jpc_head;
wire [31:0] NPC;
wire [31:0] PC;
wire reg_detect_confict;
wire PC_bobl;
wire JPC_en;
wire [31:0] JPC;
wire halt;

//IMEM

//MEM_WB
wire [4:0] rdc;
wire [31:0] rdd;
wire wen ;

//NPCmaker
wire [31:0] NPC_out;

//regfile
wire [4:0] rsc;
wire [4:0] rtc;
wire [31:0] rd;
wire [31:0] regfile0;
wire [31:0] regfile1;
wire [31:0] regfile2;
wire [31:0] regfile3;
wire [31:0] regfile4;
wire [31:0] regfile5;
wire [31:0] regfile6;
wire [31:0] regfile7;
wire [31:0] regfile8;
wire [31:0] regfile9;
wire [31:0] regfile10;
wire [31:0] regfile11;
wire [31:0] regfile12;
wire [31:0] regfile13;
wire [31:0] regfile14;
wire [31:0] regfile15;
wire [31:0] regfile16;
wire [31:0] regfile17;
wire [31:0] regfile18;
wire [31:0] regfile19;
wire [31:0] regfile20;
wire [31:0] regfile21;
wire [31:0] regfile22;
wire [31:0] regfile23;
wire [31:0] regfile24;
wire [31:0] regfile25;
wire [31:0] regfile26;
wire [31:0] regfile27;
wire [31:0] regfile28;
wire [31:0] regfile29;
wire [31:0] regfile30;
wire [31:0] regfile31;

sccomp_dataflow uut (
    .clk                (clk),
    .reset              (reset),
    .a                 (a),
    .b                 (b),
    .aluc              (aluc),
    .aluo              (aluo),
    .zero              (zero),
    .carry              (carry),
    .negative          (negative),
    .overflow           (overflow),
    .rs                (rs),
    .rt                (rt),
    .instr             (instr),
    .NPC_if_id         (NPC_if_id),
    .B_PC_en           (B_PC_en)
);

```

```

.B_PC          (B_PC           ),
.SC            (SC             ),
.LC            (LC             ),
.Data_in       (Data_in        ),
.DMEMAddr     (DMEMAddr       ),
.CS            (CS             ),
.DM_W          (DM_W           ),
.DM_R          (DM_R           ),
.Dataout       (Dataout        ),
.doing_op_id_ex (doing_op_id_ex),
.instr_id_ex  (instr_id_ex   ),
.aluo_ex_mem  (aluo_ex_mem   ),
.b_ex_mem     (b_ex_mem       ),
.instr_ex_mem (instr_ex_mem  ),
.doing_op_ex_mem (doing_op_ex_mem),
.instr_if_id  (instr_if_id   ),
.doing_op     (doing_op       ),
.jpc_head     (jpc_head       ),
.NPC          (NPC            ),
.PC            (PC             ),
.reg_detect_conflict (reg_detect_conflict),
.PC_bobl      (PC_bobl        ),
.JPC_en        (JPC_en         ),
.JPC          (JPC            ),
.halt          (halt           ),
.rdc           (rdc            ),
.rdd           (rdd            ),
.wen           (wen            ),
.NPC_out      (NPC_out        ),
.rsc           (rsc            ),
 rtc           (rtc            ),
.rd            (rd             ),
.regfile0      (regfile0       ),
.regfile1      (regfile1       ),
.regfile2      (regfile2       ),
.regfile3      (regfile3       ),
.regfile4      (regfile4       ),
.regfile5      (regfile5       ),
.regfile6      (regfile6       ),
.regfile7      (regfile7       ),
.regfile8      (regfile8       ),
.regfile9      (regfile9       ),
.regfile10     (regfile10      ),
.regfile11     (regfile11      ),
.regfile12     (regfile12      ),
.regfile13     (regfile13      ),
.regfile14     (regfile14      ),
.regfile15     (regfile15      ),
.regfile16     (regfile16      ),
.regfile17     (regfile17      ),
.regfile18     (regfile18      ),
.regfile19     (regfile19      ),
.regfile20     (regfile20      ),
.regfile21     (regfile21      ),
.regfile22     (regfile22      ),
.regfile23     (regfile23      ),
.regfile24     (regfile24      ),
.regfile25     (regfile25      ),
.regfile26     (regfile26      ),
.regfile27     (regfile27      ),
.regfile28     (regfile28      ),
.regfile29     (regfile29      ),
.regfile30     (regfile30      ),
.regfile31     (regfile31      )
);
wire [3:0] reg_lock[31:0];
genvar i;
generate
  for (i = 0; i < 32; i = i + 1) begin : reg_lock_assign
    assign reg_lock[i] = _246tb_ex10_tb.uut.sccpu.cpu_ref.reg_lock[i];
  end
endgenerate

```

```

integer file_output;

initial
begin
    file_output = $fopen("_246tb_ex10_result.txt");
// Initialize Inputs
clk = 0;
reset = 1;
count=0;
    pc_end_count=0;

// Wait 100 ns for global reset to finish
#10;
    reset = 0;
// Add stimulus here

//#100;
//$/fclose(file_output);
end
always begin
#5;
if(instr==32'hffffffff)begin
    pc_end_count=pc_end_count+1;
end
if(pc_end_count==20) begin
    $fdisplay(file_output, "regfile0: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[0]);
    $fdisplay(file_output, "regfile1: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[1]);
    $fdisplay(file_output, "regfile2: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[2]);
    $fdisplay(file_output, "regfile3: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[3]);
    $fdisplay(file_output, "regfile4: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[4]);
    $fdisplay(file_output, "regfile5: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[5]);
    $fdisplay(file_output, "regfile6: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[6]);
    $fdisplay(file_output, "regfile7: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[7]);
    $fdisplay(file_output, "regfile8: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[8]);
    $fdisplay(file_output, "regfile9: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[9]);
    $fdisplay(file_output, "regfile10: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[10]);
    $fdisplay(file_output, "regfile11: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[11]);
    $fdisplay(file_output, "regfile12: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[12]);
    $fdisplay(file_output, "regfile13: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[13]);
    $fdisplay(file_output, "regfile14: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[14]);
    $fdisplay(file_output, "regfile15: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[15]);
    $fdisplay(file_output, "regfile16: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[16]);
    $fdisplay(file_output, "regfile17: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[17]);
    $fdisplay(file_output, "regfile18: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[18]);
    $fdisplay(file_output, "regfile19: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[19]);
    $fdisplay(file_output, "regfile20: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[20]);
    $fdisplay(file_output, "regfile21: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[21]);
    $fdisplay(file_output, "regfile22: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[22]);
    $fdisplay(file_output, "regfile23: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[23]);
    $fdisplay(file_output, "regfile24: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[24]);
    $fdisplay(file_output, "regfile25: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[25]);
    $fdisplay(file_output, "regfile26: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[26]);
    $fdisplay(file_output, "regfile27: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[27]);
    $fdisplay(file_output, "regfile28: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[28]);
    $fdisplay(file_output, "regfile29: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[29]);
    $fdisplay(file_output, "regfile30: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[30]);
    $fdisplay(file_output, "regfile31: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[31]);
$stop;
end
count=count+clk;

clk= ~clk;
// if(clk== 1'b1&&count!=0) begin
//     $fdisplay(file_output, "pc: %h", PC);
//     $fdisplay(file_output, "instr: %h", instr);
//     $fdisplay(file_output, "regfile0: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[0]);
//     $fdisplay(file_output, "regfile1: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[1]);
//     $fdisplay(file_output, "regfile2: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[2]);
//     $fdisplay(file_output, "regfile3: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[3]);

```

```

// $fdisplay(file_output, "regfile4: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[4])
// $fdisplay(file_output, "regfile5: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[5])
// $fdisplay(file_output, "regfile6: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[6])
// $fdisplay(file_output, "regfile7: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[7])
// $fdisplay(file_output, "regfile8: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[8])
// $fdisplay(file_output, "regfile9: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[9])
// $fdisplay(file_output, "regfile10: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[10])
// $fdisplay(file_output, "regfile11: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[11])
// $fdisplay(file_output, "regfile12: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[12])
// $fdisplay(file_output, "regfile13: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[13])
// $fdisplay(file_output, "regfile14: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[14])
// $fdisplay(file_output, "regfile15: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[15])
// $fdisplay(file_output, "regfile16: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[16])
// $fdisplay(file_output, "regfile17: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[17])
// $fdisplay(file_output, "regfile18: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[18])
// $fdisplay(file_output, "regfile19: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[19])
// $fdisplay(file_output, "regfile20: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[20])
// $fdisplay(file_output, "regfile21: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[21])
// $fdisplay(file_output, "regfile22: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[22])
// $fdisplay(file_output, "regfile23: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[23])
// $fdisplay(file_output, "regfile24: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[24])
// $fdisplay(file_output, "regfile25: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[25])
// $fdisplay(file_output, "regfile26: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[26])
// $fdisplay(file_output, "regfile27: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[27])
// $fdisplay(file_output, "regfile28: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[28])
// $fdisplay(file_output, "regfile29: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[29])
// $fdisplay(file_output, "regfile30: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[30])
// $fdisplay(file_output, "regfile31: %h", _246tb_ex10_tb.uut.sccpu.cpu_ref.array_reg[31])

// end
end

endmodule

```

### 11.2.2 自动化测试脚本 (run\_cpu\_tests.do)

```
# ModelSim Batch Script for CPU Pipeline Testing

# Clean up any existing libraries and files
if {[file exists work]} {
    vdel -lib work -all
}
# Don't delete transcript and vsim.wlf as they might be in use
# Clean up any old result files if they exist
catch {file delete _246tb_ex10_result.txt}

# Create fresh working library
vlib work

# Compile IP core files first (these are required for the design to work)
vlog -work work -quiet ./cpupip8.srcts/sources_1/ip/dmem1/dist_mem_gen_v8_0_10/simulation/dist_mem_gen_v8_0_10.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/ip/dmem1/sim/dmem1.v

# Compile source files
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/def.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/alu.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/BJudge.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/PCreg.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/DMEM.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/EX_MEM.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/ID_EX.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/IF_ID.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/IMEM.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/MEM_WB.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/NPCmaker.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/regfile.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/cpu.v
vlog -work work -quiet ./cpupip8.srcts/sources_1/new/sccomp_dataflow.v
vlog -work work -quiet ./cpupip8.srcts/sim_1/new/_246tb_ex10_tb.v

# Create results directory using Tcl commands
set results_dir "./test_scripts/results"
if {! [file exists $results_dir]} {
    file mkdir $results_dir
}

# List of all test files to run
set test_files [list \
    "testdata/1_addi.hex.txt" \
    "testdata/2_addiu.hex.txt" \
    "testdata/9_addu.hex.txt" \
    "testdata/11_beq.hex.txt" \
    "testdata/12_bne.hex.txt" \
    "testdata/16_26_lsws.hex.txt" \
    "testdata/16_26_lsws2.hex.txt" \
    "testdata/20_sll.hex.txt" \
    "testdata/22_sltu.hex.txt" \
    "testdata/25_subu.hex.txt" \
    "testdata/101_swlbnebeq.hex.txt" \
    "testdata/102_regconflict.hex.txt" \
    "testdata/103_regconflict_detected_2.hex.txt" \
    "testdata/104_pizza_tower_test.hex.txt" \
]

# Procedure to run a single test
proc run_single_test {test_file results_dir} {
    # Get test name without extension
    set test_name [file rootname [file tail $test_file]]
    set test_name [string map {(.hex "")} $test_name]

    puts "\n-----"
    puts "RUNNING TEST: $test_file"
    puts "-----"

    # Clean up any old result files before starting the test
    if {[file exists ./_246tb_ex10_result.txt]} {
```

```

        catch {file delete ./_246tb_ex10_result.txt}
    }

# Backup original IMEM.v
file copy -force ./cpupip8.srcs/sources_1/new/IMEM.v ./cpupip8.srcs/sources_1/new/IMEM.v.bak

# Read the original IMEM.v
set fid [open "./cpupip8.srcs/sources_1/new/IMEM.v" r]
set content [read $fid]
close $fid

# Replace the readmemh line with the current test file
set lines [split $content "\n"]
set new_lines {}
foreach line $lines {
    if {[string match "*\$readmemh*" $line] && [string match "*IMEMreg*" $line] && ![string match "# Found the active $readmemh line, replace it" $line]} {
        lappend new_lines "\$readmemh(\"E:/Homeworks/cpupip8/$test_file\", IMEMreg);"
    } else {
        lappend new_lines $line
    }
}

set new_content [join $new_lines "\n"]

# Write the modified IMEM.v
set fid [open "./cpupip8.srcs/sources_1/new/IMEM.v" w]
puts -nonewline $fid $new_content
close $fid

# Recompile only IMEM.v
vlog -work work -quiet ./cpupip8.srcs/sources_1/new/IMEM.v

# Load and run simulation
vsim -quiet work._246tb_ex10_tb

# Run simulation for maximum 100000ns, but check for halt condition
run 100000ns

# Copy the test result file to the results directory
set sim_result_file "./_246tb_ex10_result.txt"
set output_result_file "$results_dir/${test_name}_sim_result.txt"

if {[file exists $sim_result_file]} {
    file copy -force $sim_result_file $output_result_file
    puts "Saved simulation result to: $output_result_file"
} else {
    puts "Warning: Simulation result file not found: $sim_result_file"
}

# Restore original IMEM.v
file copy -force ./cpupip8.srcs/sources_1/new/IMEM.v.bak ./cpupip8.srcs/sources_1/new/IMEM.v
file delete ./cpupip8.srcs/sources_1/new/IMEM.v.bak

# Use existing standard result file from testdata/
set original_std_result_file "E:/Homeworks/cpupip8/$test_file"
regsub {\.hex\.txt\$} $original_std_result_file ".result.txt" std_result_file

# Copy standard result file to the results directory
set output_std_result_file "$results_dir/${test_name}_std_result.txt"
if {[file exists $std_result_file]} {
    file copy -force $std_result_file $output_std_result_file
    puts "Standard test result copied to: $output_std_result_file"
} else {
    puts "Standard result file not found: $std_result_file"
    return 0
}

# Compare simulation result with standard result using external tool
set compare_result [catch {exec txt_compare --file1 $output_result_file --file2 $std_result_file}]

# Check the comparison result

```

```

set comp_file "$results_dir/${test_name}_comparison_result.txt"
if {[file exists $comp_file]} {
    set comp_fid [open $comp_file r]
    set comp_content [read $comp_fid]
    close $comp_fid

    # 检查是否包含成功的输出行
    set lines [split $comp_content "\n"]
    set success 0
    foreach line $lines {
        # 检查是否包含"在指定检查条件下完全一致."
        if {[string match "*在指定检查条件下完全一致.*" $line]} {
            set success 1
            break
        }
    }

    if {$success} {
        puts "RESULT: PASS - $test_file"
        return 1
    } else {
        puts "RESULT: FAIL - $test_file"

        # 输出失败的具体信息
        puts "Comparison output:"
        puts "===="
        puts $comp_content
        puts "===="

        return 0
    }
} else {
    puts "Comparison result file not found: $comp_file"
    return 0
}
}

# Run all tests
set total_tests [llength $test_files]
set pass_count 0

puts "Starting batch simulation for $total_tests tests..."

foreach test_file $test_files {
    set result [run_single_test $test_file $results_dir]
    if {$result == 1} {
        incr pass_count
    }
}

# Generate final summary
set summary_text [subst "BATCH TEST SUMMARY\n=====\\nTotal tests: $total_tests\\n$summary_text"]

puts "\\n$summary_text"

# Write summary to file
set sum_fid [open "$results_dir/test_summary.txt" w]
puts $sum_fid $summary_text
close $sum_fid

puts "\\nAll tests completed. Results saved in $results_dir/"
puts "Success rate: [format %.2f [expr double($pass_count)*100/double($total_tests)]]%"

# Quit ModelSim
quit

```

### 三. 性能验证模型

#### 11.3.1 比萨塔摔鸡蛋游戏验证模型 (pizza\_tower.asm)

```
# 比萨塔摔鸡蛋游戏验证模型
# 寄存器使用说明：
# $s0: 总层数N
# $s1: 耐摔值F
# $s2: 当前鸡蛋数
# $s3: 总摔次数
# $s4: 总上楼数m
# $s5: 总下楼数n
# $s6: 摔破鸡蛋数h
# $s7: 最后摔的状态 (0:未破, 1:已破)
# $t0-$t9: 临时寄存器

# 内存布局：
# 0x0000: 输入N (总层数)
# 0x0004: 输入F (耐摔值)
# 0x0008: 输出总摔次数
# 0x000C: 输出总鸡蛋数
# 0x0010: 输出最后状态
# 0x0014: 输出成本1 (物质匮乏时期)
# 0x0018: 输出成本2 (人力成本增长时期)

# 初始化
main:
    # 加载输入参数
    lw $s0, 0x0000($zero)      # 加载总层数N
    lw $s1, 0x0004($zero)      # 加载耐摔值F

    # 初始化变量
    addiu $s2, $zero, 1          # 鸡蛋数从1开始
    addiu $s3, $zero, 0          # 总摔次数初始为0
    addiu $s4, $zero, 0          # 总上楼数初始为0
    addiu $s5, $zero, 0          # 总下楼数初始为0
    addiu $s6, $zero, 0          # 摔破鸡蛋数初始为0
    addiu $s7, $zero, 0          # 最后状态初始为0

    # 二分查找算法
    addiu $t0, $zero, 1          # low = 1
    add $t1, $zero, $s0           # high = N
    addiu $t2, $zero, 0          # 当前楼层
    addiu $t3, $zero, 1          # 鸡蛋完好标志

search_loop:
    # 检查搜索是否结束
    sltu $t4, $t0, $t1           # 比较low < high
    beq $t4, $zero, search_end

    # 计算中间楼层
    add $t2, $t0, $t1             # low + high
    sll $t5, $t2, 31              # 判断奇偶
    beq $t5, $zero, even_mid
    addiu $t2, $t2, 1              # 奇数加1
even_mid:
    srl $t2, $t2, 1                # 除以2得到中间楼层

    # 摔鸡蛋测试
    addiu $s3, $s3, 1              # 总摔次数+1

    # 计算上楼数
    subu $t6, $t2, $t0             # 当前楼层 - low
    add $s4, $s4, $t6              # 累加上楼数

    # 测试鸡蛋
    sltu $t7, $t2, $s1             # 比较楼层 < 耐摔值
    addiu $t8, $s1, 1                # 比较楼层 <= 耐摔值
    sltu $t9, $t2, $t8              # 比较楼层 <= 耐摔值
```

```

beq $t9, $zero, egg_broken

# 鸡蛋未破
addiu $s7, $zero, 0          # 最后状态=0 (未破)
add $t0, $zero, $t2          # low = mid
addiu $t0, $t0, 1            # low = mid + 1

# 计算下楼数
subu $t6, $t1, $t2          # high - 当前楼层
add $s5, $s5, $t6             # 累加下楼数

beq $zero, $zero, search_continue

egg_broken:
# 鸡蛋摔破
addiu $s6, $s6, 1            # 摔破鸡蛋数+1
addiu $s2, $s2, 1            # 总鸡蛋数+1 (使用新鸡蛋)
addiu $s7, $zero, 1           # 最后状态=1 (已破)
add $t1, $zero, $t2           # high = mid

# 计算下楼数
subu $t6, $t2, $t0           # 当前楼层 - low
add $s5, $s5, $t6             # 累加下楼数

search_continue:
beq $zero, $zero, search_loop

search_end:
# 最后一次测试
addiu $s3, $s3, 1            # 总摔次数+1

# 测试鸡蛋
sltu $t7, $t0, $s1           # 比较楼层 < 耐摔值
addiu $t8, $s1, 1
sltu $t9, $t0, $t8           # 比较楼层 <= 耐摔值
beq $t9, $zero, final_broken

# 鸡蛋未破
addiu $s7, $zero, 0           # 最后状态=0
beq $zero, $zero, save_results

final_broken:
# 鸡蛋摔破
addiu $s6, $s6, 1            # 摔破鸡蛋数+1
addiu $s2, $s2, 1            # 总鸡蛋数+1
addiu $s7, $zero, 1           # 最后状态=1

save_results:
# 保存输出结果
sw $s3, 0x0008($zero)        # 保存总摔次数
sw $s2, 0x000C($zero)        # 保存总鸡蛋数
sw $s7, 0x0010($zero)        # 保存最后状态

# 计算成本1 (物质匮乏时期: p1=2, p2=1, p3=4)
# cost1 = m*2 + n*1 + h*4
sll $t0, $s4, 1              # m * 2
add $t1, $s5, $zero           # n * 1
sll $t2, $s6, 2              # h * 4
add $t3, $t0, $t1
add $t3, $t3, $t2             # 总成本1
sw $t3, 0x0014($zero)        # 保存成本1

# 计算成本2 (人力成本增长时期: p1=4, p2=1, p3=2)
# cost2 = m*4 + n*1 + h*2
sll $t0, $s4, 2              # m * 4
add $t1, $s5, $zero           # n * 1
sll $t2, $s6, 1              # h * 2
add $t3, $t0, $t1
add $t3, $t3, $t2             # 总成本2
sw $t3, 0x0018($zero)        # 保存成本2

```

```
# 程序结束  
halt  
# 程序结束
```

## 四. FPGA 约束文件

### 11.4.1 XDC 约束文件 (cpupip8.xdc)

```
set_property PACKAGE_PIN E3 [get_ports clk]
set_property PACKAGE_PIN N17 [get_ports rst]
set_property PACKAGE_PIN V10 [get_ports ena]
set_property PACKAGE_PIN M13 [get_ports switch[2]]
set_property PACKAGE_PIN L16 [get_ports switch[1]]
set_property PACKAGE_PIN J15 [get_ports switch[0]]

set_property PACKAGE_PIN T10 [get_ports {o_seg[0]}]
set_property PACKAGE_PIN R10 [get_ports {o_seg[1]}]
set_property PACKAGE_PIN K16 [get_ports {o_seg[2]}]
set_property PACKAGE_PIN K13 [get_ports {o_seg[3]}]
set_property PACKAGE_PIN P15 [get_ports {o_seg[4]}]
set_property PACKAGE_PIN T11 [get_ports {o_seg[5]}]
set_property PACKAGE_PIN L18 [get_ports {o_seg[6]}]
set_property PACKAGE_PIN H15 [get_ports {o_seg[7]}]

set_property PACKAGE_PIN J17 [get_ports {o_sel[0]}]
set_property PACKAGE_PIN J18 [get_ports {o_sel[1]}]
set_property PACKAGE_PIN T9 [get_ports {o_sel[2]}]
set_property PACKAGE_PIN J14 [get_ports {o_sel[3]}]
set_property PACKAGE_PIN P14 [get_ports {o_sel[4]}]
set_property PACKAGE_PIN T14 [get_ports {o_sel[5]}]
set_property PACKAGE_PIN K2 [get_ports {o_sel[6]}]
set_property PACKAGE_PIN U13 [get_ports {o_sel[7]}]

set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports rst]
set_property IOSTANDARD LVCMOS33 [get_ports ena]
set_property IOSTANDARD LVCMOS33 [get_ports switch[2]]
set_property IOSTANDARD LVCMOS33 [get_ports switch[1]]
set_property IOSTANDARD LVCMOS33 [get_ports switch[0]]

set_property IOSTANDARD LVCMOS33 [get_ports {o_sel[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_sel[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_sel[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_sel[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_sel[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_sel[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_sel[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_sel[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {o_seg[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_seg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_seg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_seg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_seg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_seg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_seg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_seg[0]}]

create_clock -period 100.000 -name clk_pin -waveform {0.000 50.000} [get_ports clk]
set_input_delay -clock [get_clocks *] 1.000 [get_ports reset]
set_output_delay -clock [get_clocks *] 0.000 [get_ports -filter { NAME =~ "*" && DIRECTION == "OUT" }]
```