# 1 The MIPS R1000 SuperScalar MircoProcessor

**MIPS R10000** 是一个动态超标量处理器，实现了 64-bit 的 MIPS 4 指令集。每个周期取指并且译码 4 条指令，分支指令预测执行。R10000 乱序执行，支持内存一致性和精确异常。其 Cache 设计采用两路组相连，非阻塞方式，写回策略，隐藏访问内存的延迟。R10000 采用模块化设计的方式，在控制逻辑上使用常规结构：激活列表，寄存器映射表以及指令队列。

## Design rationale

内存带宽以及延迟限制了程序的性能：R10000 使用寄存器映射和非阻塞 Cache 设计降低延迟。非阻塞 Cache 指在 Cache 写回或者是某一行重填的的过程中，不阻塞其他 Cache 行的访问。编译器可以根据 Cache 的特点更好的优化代码，浮点应用的优化最为明显。R10000 动态的重新排序指令执行，处理器向前查找 32 条指令以查找可能的并行，提交阶段恢复指令顺序。

## Implementation

- 四路超标量处理器，每周期取指译码 4 条指令；
- 支指令预测执行，包含有 4 项分支栈（用于推测错误时，CPU 现场的快速恢复）；
- 采用动态乱序执行；
- 使用映射表实现寄存器重命名；
- 支持精确异常；
- 一个非阻塞 load/store 单元；
- 双 64-bit 整数运算单元；
- 64-bit，IEEE Std 754-1985 浮点运算单元；
- 流水线乘法器，2 周期延迟；
- 流水线地址运算器，2 周期延迟；
- 一级私有 Cache：32 KB 指令 Cache，32KB 的数据 Cache，采取两路组相连；
- 二级片外 Cache：128 bit 位宽，两路组相连；
- 64 bit 的接口用于多处理器互联；
- 其中二级 Cache 的大小为：512KB-16MB；

## Instruction fetch

每次预取 4 条指令，修改一级 Cache 的结构，可以非对齐访问一级 Cache，但是没有说明预取得 4 条指令如何跨 Cache 行访问;

## Branch unit

2 级自适用预测算法，没有具体阐述分支预测器的设计细节；
4 项 branch stack，可以连续进行 4 次分支预测，branch stack 保存分支预测时 CPU 的现场环境，用于预测失败时的快速恢复

## Decode logic

在激活列表以及指令队列不满的情况下，可以同时译码 4 条指令；长乘法以及除法占据激活列表的两项，在译码该指令时，其后续指令都不可以参与译码过程

## Register mapping

**预译码阶段.** 调整指令格式，增加 4-bit 单位字段

- 译码阶段：浮点比较指令设置状态寄存器的 8 个条件位；
- 重命名策略：基于映射表，动态映射物理寄存器和逻辑寄存器；
- 定点寄存器：33 个逻辑寄存器（r1-r31, Lo and Hi），64 个物理整数寄存器；
- 浮点寄存器：32 个逻辑寄存器，64 个物理浮点寄存器；

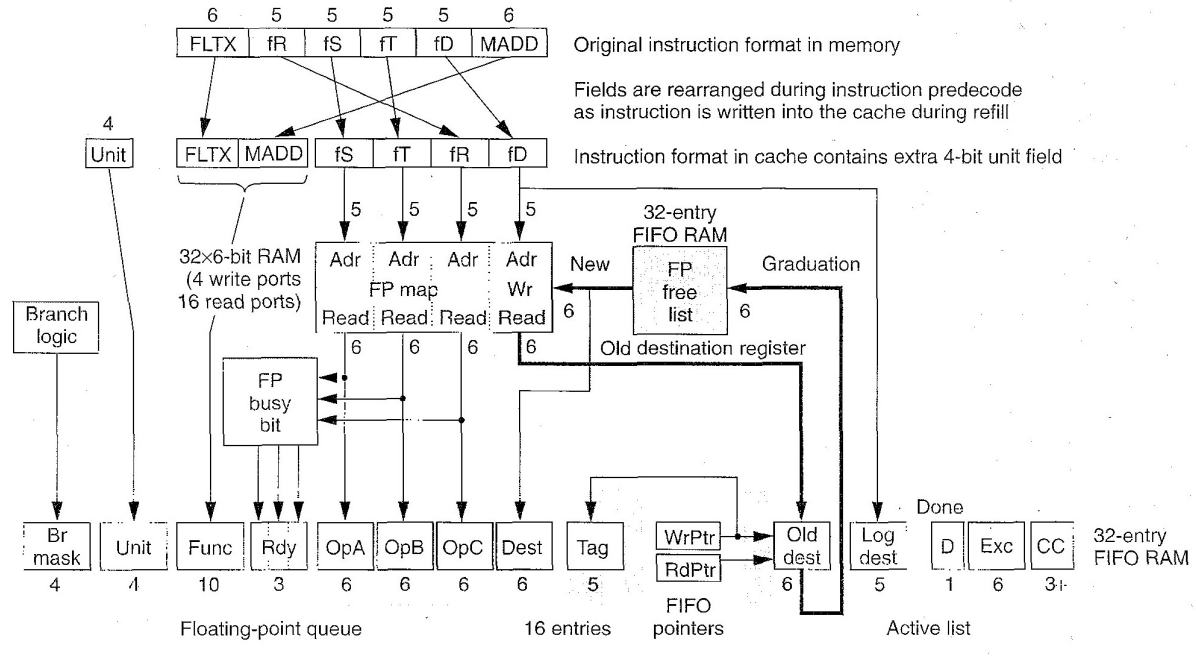


Figure 5. Register renaming, pipeline stage 2. The R10000 rearranges fields during instruction predecode as it writes the instruction into the cache during refill. The instruction format in the cache contains an extra 4-bit unit field.

**Register map tables.** 以浮点部件为例：16 x 5-bit 读端口（其中一个读端口被 Old destination register 使用），4 x 6-bit 写端口，可以同时映射 4 条指令；24 five-bit 比较器决定取指的 4 指令之间的数据依赖关系，决定旁路复用

**Free list.** 整数和浮点数 free list 包含未被使用的物理寄存器，当 register map table 进行重映射时，从 free list 中选取空闲物理寄存器，与逻辑寄存器进行关联。当一个 old destination register 中的物理寄存器不在被后续指令使用时，该物理寄存器返回 free list；

**Active List.** Active list 记录当前处于激活状态（指的“在发射队列的指令”）的指令，最多可以同时记录 32 条激活指令，组织成为 4 路并行，深度为 8 的 FIFO；每条指令都有 5-bit tag，当执行单元完成时，将置位激活列表中的完成位；激活列表的一条记录包含该指令的逻辑寄存器以及其所对应的旧的物理寄存器，用于中断时 CPU 现场的快速恢复

**Busy-bit tables.** 64x1-bit 多端口 RAM 处理，处理 RAW 数据相关，当关联物理寄存器离开 free list 时，将该物理寄存器置位为忙，意味着该物理寄存器目前不可读；当执行阶段完成后，该位置为空闲，意味着该物理寄存器可以被后续指令读

## Instruction queues

Integer queue（16 entries）ALU1 中，分支和移位指令具有高优先级；ALU2 中乘法和除法指令具有高优先级；Integer queue 可以处理队列指令之间存在的相关性；
FP queue（16 entries）同 Integer queue 的功能类似，但是不包含直接数；
Address queue（16 entries）其指令优先级是由进栈的顺序决定的，组织为一个 FIFO。使用 2 个 16x16 的矩阵跟踪存储器访问之间的依赖关系。MIPS 中的 LL，SC 指令并不锁定对内存的访问，降低了访存的复杂性

## Register files

整数寄存器文件有七个读端口和三个写端口。这些包括两个专用读端口和一个用于每个 ALU 的专用写端口以及两个用于地址计算单元的专用读端口。整数寄存器的第七个读端口处理存储，跳转寄存器和移动点浮点指令。它的第三个写端口处理加载，分支和链接以及从浮点移动指令。
浮点寄存器文件有五个读端口和三个写端口。加法器和乘法器各有两个专用读端口和一个专用写端口。第五个读端口处理存储和移动指令; 第三个写端口处理加载和移动指令

## Integer execution units

Integer ALU：两个整数 ALU 中的每一个包含 64 位加法器和逻辑单元。ALU 1 包含 64 位移位器和分支条件逻辑，ALU 2 包含部分整数乘法器数组和整数除法逻辑
Integer multiplication and division. 乘法 Booth’s algorithm，除法 nonrestoring algorithm

## Floating-point execution units

浮点运算单元主要由三部分组成：浮点加法单元；浮点乘法单元；浮点除法以及开根号单元，浮点指令的延迟如 Table 2 所示h

**Table 2. Latency and repeat rates for floating-point instructions.**

| Unit | Latency (cycles) | Repeat rate (cycles) | Instruction |
|---|---|---|---|
| Add | 2 | 1 | Add, subtract, compare |
| Multiply | 2 | 1 | Integer branches |
| Divide | 12 | 14 | 32-bit divide |
| | 19 | 21 | 64-bit divide |
| Square root | 18 | 20 | 32-bit square root |
| | 33 | 35 | 64-bit square root |
| Load/store | 3 | 1 | Load floating-point value |
| | — | 1 | Store floating-point value |

## Memory hierarchy

两级 Cache 都是用 LRU 置换算法；

Load store unit. 当缓存不忙时，加载指令同时访问 TLB，缓存标记数组和缓存数据数组。并行访问导致两个周期的加载延迟。

TLB. 64 项全项联，将 44-bit 的虚拟地址转化为 40-bit 的物理地址

## System interface

Clocks：使用 PLL 产生同步时钟信号，流水线频率 200Mhz，其他部件时钟频率由 PLL 生成

Test feature：十个 128-bit 线性反馈移位寄存器

## Performance

配置：200MHz R10000 微处理器，4-Mbyte 200MHz 二级缓存，100MHz 系统接口总线，180ns 内存延迟

SPEC95int (peak)　9

SPEC95fp (pesk)　19