KXCore 项目文档

概述

KXCore 是一个基于 la32r 指令集的乱序多发射 CPU, 支持3发射队列，双提交同时具有分支预测，Cache等优化设计。

我们实现了 LoongArch32 精简版，支持：

- **基础指令集**：包含算术运算、逻辑运算、移位、分支跳转等指令
- **访存指令**：支持字节、半字、字的加载存储操作
- **系统指令**：ERTN、SYSCALL、EBREAK等特权指令
- **CSR操作**：CSRRD、CSRWR、CSRXCHG等控制状态寄存器指令
- **中断异常处理**：完整的异常处理机制和中断控制
- **AXI总线接口**：支持标准AXI3总线协议

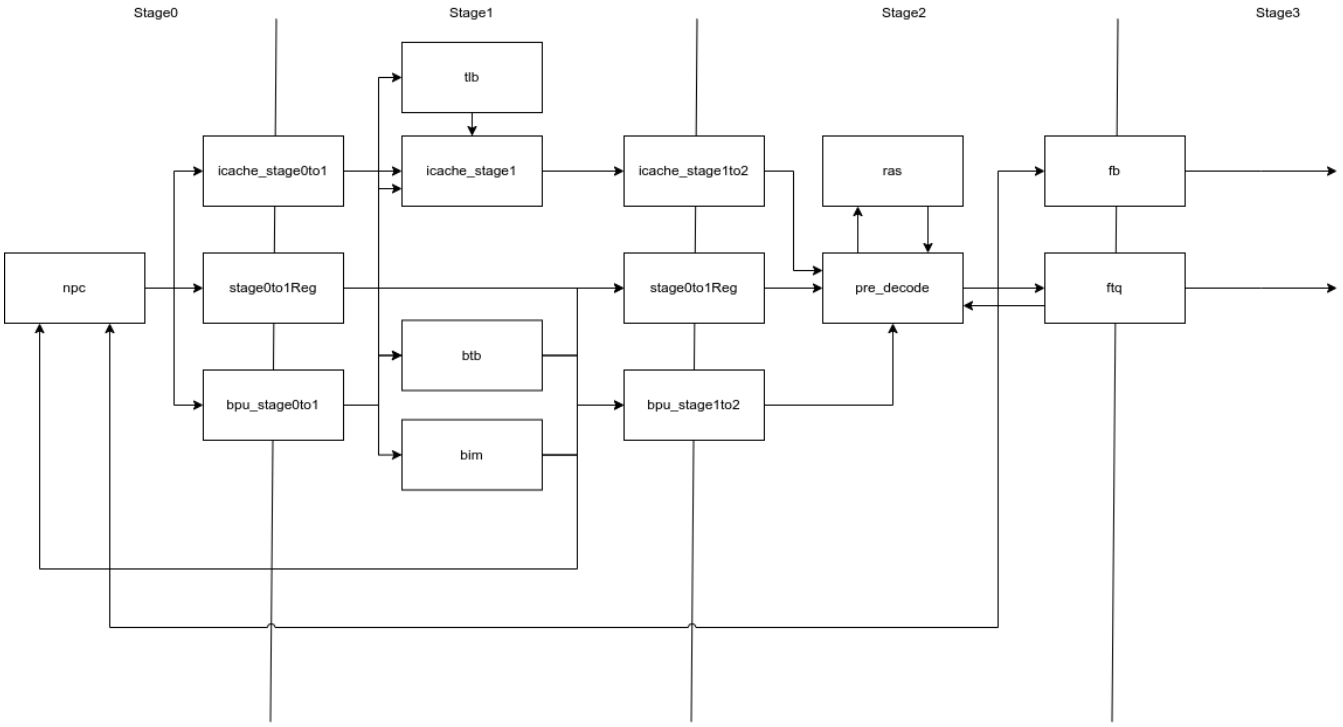
核心参数

- **数据位宽**：32位
- **取指宽度**：四路取指 (fetchWidth = 4)
- **译码宽度**：
- **发射宽度**：三发射单元，四执行单元，最大允许发射四条指令
- **退休宽度**：单周期退休 (retireWidth = 1)
- **物理寄存器**：80个物理寄存器，32个逻辑寄存器
- **ROB大小**：32个条目的重排序缓冲区

总体架构

KXCore 使用了前后端架构，前端4阶段，后端6阶段。

前端设计



前端采用4级流水线设计，包含以下阶段：

Stage 0: 预取指阶段

- **PC生成**：支持顺序取指和重定向（后端/预测单元预测结果）

Stage 1: 分支预测与取指阶段

- **ICache**：指令缓存，加快前端取指速度
- **TLB**:翻译虚拟地址
- **分支预测**：通过BTB、BIM单元预测分支结果，帮助PC进行重定向

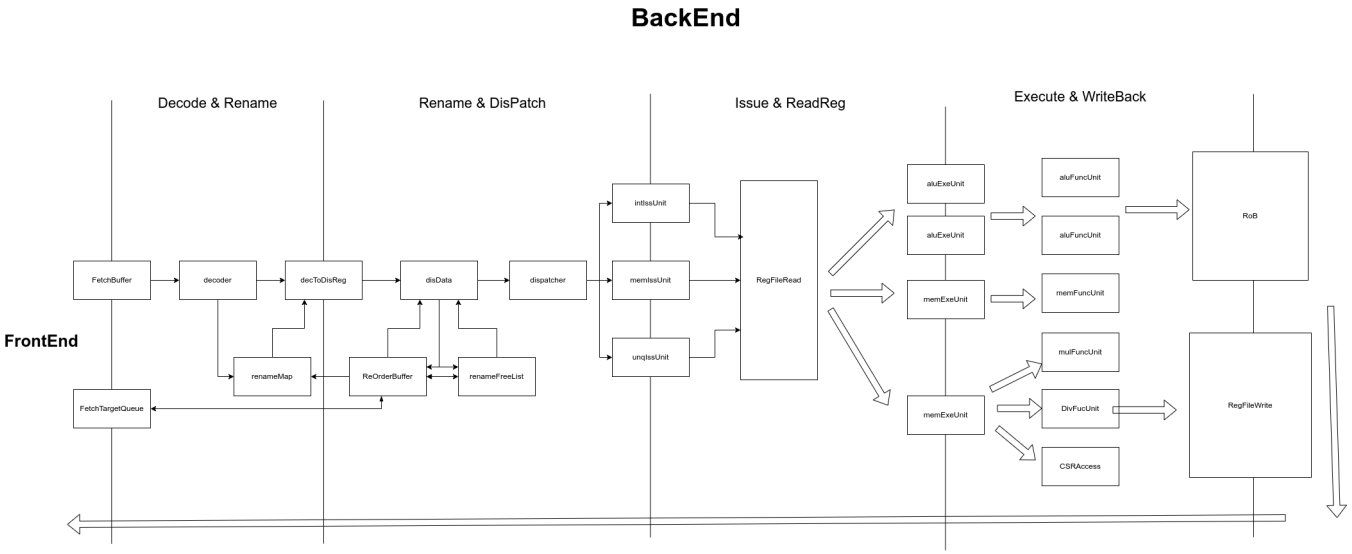
Stage 2: 预解码阶段

- **RAS**：函数调用地址堆栈
- **预解码**:提前完成控制流指令的解码，帮助PC进行重定向

Stage 3: 发送指令阶段

- **取指缓冲区 (FetchBuffer)**：16个条目，用于缓冲取指结果
- **取指目标队列 (FTQ)**：32个条目，缓存取指地址和分支信息，减少后端流水线寄存器存储压力

后端设计



Stage 1: 译码与重命名1阶段 (`Decoder`)

- **指令译码**：将指令译码为微操作(MicroOp)
- **操作数识别**：识别源寄存器和目标寄存器
- **功能单元分配**：确定指令需要的执行单元类型
- **寄存器重命名**：逻辑寄存器到物理寄存器的映射

Stage 2: 重命名2与分派阶段 (`Dispatch`)

- **依赖关系解析**：建立指令间的数据依赖关系
- **物理寄存器分配**：从空闲列表分配物理寄存器
- **ROB分配**：为指令分配重排序缓冲区条目
- **发射队列选择**：根据指令类型分发到对应发射队列
- **资源可用性检查**：确保有足够资源处理指令

Stage 3: 发射阶段

KXCore配置了3个专用发射队列：

1. **内存发射队列 (MEM IQ)**：
    - 处理所有访存指令 (LD/ST)
    - 发射宽度：1，队列深度：12
2. **特殊发射队列 (UNQ IQ)**：
    - 处理特殊指令 (CSR操作、系统指令等)
    - 发射宽度：1，队列深度：8
3. **整数发射队列 (INT IQ)**：
    - 处理整数运算指令 (ALU操作)
    - 发射宽度：2，队列深度：8

Stage 4: 执行阶段

- **ALU单元**：2个ALU执行单元，支持并行计算
- **访存单元**：集成DCache的内存执行单元
- **乘法器**：3级流水线乘法器
- **除法器**：支持除法运算的专用单元

Stage 5: 写回阶段

- **结果写回**：执行结果写回物理寄存器文件
- **唤醒机制**：唤醒等待的相关指令

Stage 6: 提交阶段

- **按序提交**：保证指令按程序顺序提交
- **异常处理**：处理异常和中断
- **分支重定向**：处理分支预测错误
- **架构状态更新**：更新架构可见的处理器状态

关键优化设计

分支预测机制

- **分支指令缓冲区 (BIM)**：2048组的分支历史表
- **分支目标缓冲区 (BTB)**：128组2路组相联
- **返回地址栈 (RAS)**：32条目的函数调用栈

缓存系统

- **指令缓存 (ICache)**：支持指令预取和缓存一致性

乱序执行特性

- **物理寄存器重命名**：80个物理寄存器消除WAR和WAW冲突
- **动态调度**：基于数据依赖的乱序发射
- **推测执行**：支持分支预测的推测执行
- **精确异常**：通过ROB保证精确异常处理