MeowCPU 设计报告

北京邮电大学　飞飞带飞队
任飞　梁宸　赵一鸣　申梦飞

目录

1 概述

1.1 项目背景

本项目是第七届“龙芯杯”全国大学生计算机系统能力培养大赛（NSCSCC 2023）团体赛 LoongArch 指令集赛道的参赛作品。本项目在龙芯的 FPGA 开发板上实现了基于 LoongArch32-Reduced 指令集的顺序双发射 CPU —— MeowCPU。本项目可以通过比赛提供的所有功能测试和性能测试，可以通过 PMON 引导启动 Linux 操作系统。

1.2 开发环境

本项目使用 SystemVerilog 语言开发。借助 Chiplab 项目提供的 difftest 框架进行快速仿真和调试，使用 Vivado 进行综合和实现。

1.3 性能得分

本 CPU 在性能测试中的频率为 90MHz，IPC 比值为 1.275。

1.4 设计修改

决赛作品在初赛作品的基础上添加了 TLB 和其他启动系统所需要的指令。TLB 的加入对 CPU 的性能造成了一定的影响。

2 CPU 设计方案

2.1 指令集支持

MeowCPU 支持 LoongArch32-Reduced 中除基础浮点数指令集外的所有指令。其中 `PRELD` 指令实现为 `NOP`，不会实际进行预加载操作。另外目前没有实现用户态下的特权指令访问限制。

2.2 流水线架构

CPU 的整体架构图如下。

本项目的设计路线为顺序双发射七级流水线 CPU，设计目标是尽可能挖掘指令双发的机会以及降低访存延迟。除了操作 CSR、TLB 等的特殊指令外，普通指令可在两条流水线的任意一条上被发射。

第二条流水线具有延迟 ALU 设计，当两条指令均为 ALU 指令且存在 RAW 冲突时，第二条指令可以使用第一条指令的计算结果作为操作数。

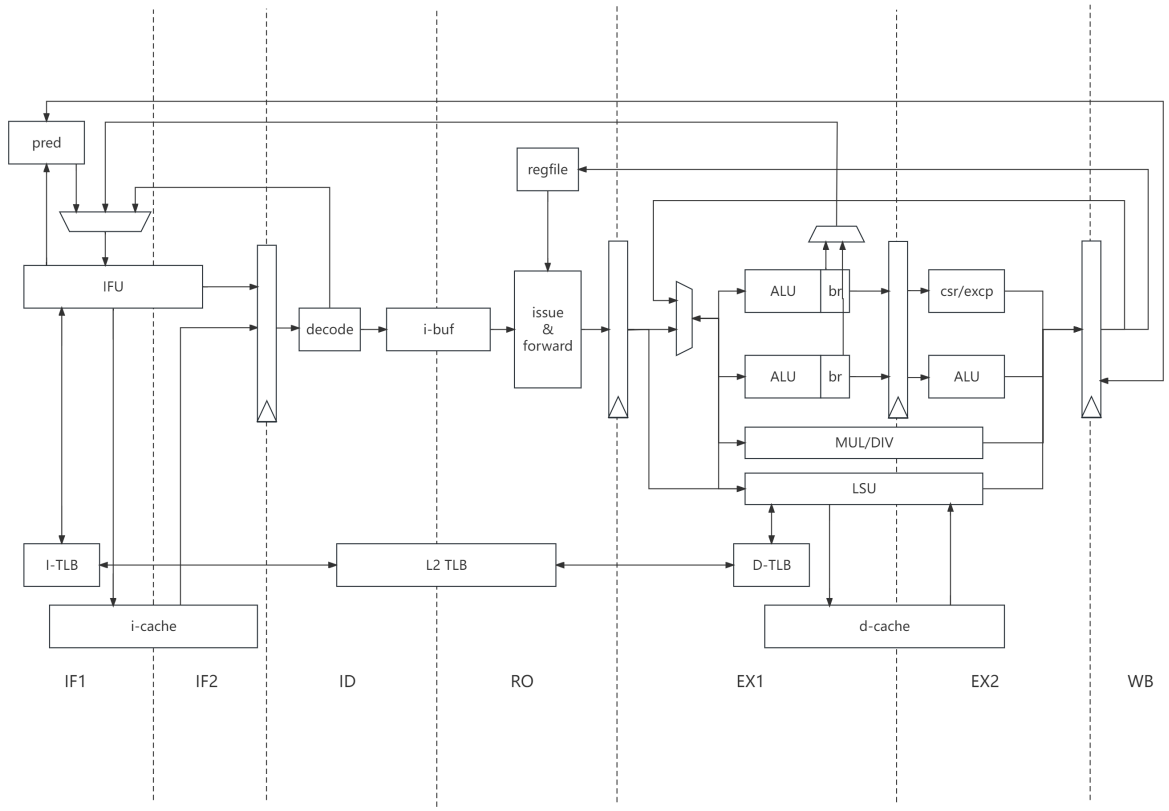

图 1: CPU 整体架构图

对于来自 ALU 的运算结果，直接前递到 RO 阶段；对于来自访存的运算结果，在 RO 阶段进行预测并前递到 EX1 阶段。值得注意的是，出于时序考虑，访存指令的基地址寄存器不允许在 EX1 阶段前递。

下面介绍各个阶段。

2.2.1 IF1 取指 1

向 I-Cache 发出取指请求，同时进行分支预测。

2.2.2 IF2 取指 2

从 I-Cache 取出指令。

2.2.3 ID 译码

进行译码并存入指令队列，同时处理无条件直接跳转。

2.2.4 RO 读操作数

从指令队列中取出指令，进行双发射条件检查和前递。如果操作数在寄存器堆、WB 阶段寄存器或是某个 ALU 的运算结果，则在此阶段直接读取操作数。当操作数来自第一条流水线即将完成的 load 指令时，标记该操作数从 WB 阶段寄存器获取。此外，如果两条指令都是 ALU 运算且第二条指令依赖第一条指令，会将第二条指令标记为延迟执行。

2.2.5 EX1 执行 1

根据 RO 阶段的标记读取操作数，并根据指令类型使用相应的执行部件。执行部件包括 3 个 ALU（其中一个在 EX2 阶段）、2 个乘法器、1 个除法器和 2 个访存单元。ALU 同时负责跳转，在该阶段处理除无条件直接跳转以外的所有跳转指令。

如果 D-Cache 或者 D-TLB 需要被用到但没有准备好，则在此阶段停顿。

2.2.6 EX2 执行 2

对于标记为延迟执行的第二条 ALU 指令，使用 EX1 阶段第一条指令的计算结果作为操作数计算。

在该阶段等待两条指令均完成。乘法运算需要 2 个周期，故在 EX2 需要停顿一个周期；除法运算共需要 34 个周期，故在 EX2 需要停顿 33 个周期；访存指令在 Cache 命中的情况下不需要停顿，未命中则需要等待总线响应。

在该阶段还进行 CSR 操作和异常处理。

2.2.7 WB 写回

将计算结果写回寄存器堆，更新分支方向预测器。

当一条指令提前完成时会将 WB 阶段寄存器作为缓存等待另一条指令完成。

2.3 指令队列设计

指令队列将 CPU 划分为前端和后端，主要功能有：

1. 解决前后端在不同时刻停顿导致速度不匹配的问题，减少取指跟不上或后端阻塞着浪费取指机会的情况。
2. 凑出两条指令一起发射。

根据实验，指令队列设置为 16 项可以获得比较理想的效果。实现上为了优化时序和减少资源占用使用了双 bank 交替的实现方法。

2.4 Cache 设计

Cache 采用非阻塞式设计，使用一个状态机进行读写状态控制，具有可开关的关键字优先功能，具备提前重启功能，具有一行写缓存。采用 VIPT 设计，提前读取元信息。data 使用 bram 存储，valid 使用寄存器存储，其他元信息使用 dram 存储。icache 独立设计了可开关的下一行预取功能，具备同行一次性读出连续两条指令的能力。

dcache 独立设计了第二辅助端口，第二端口为第一端口的辅助口，可同时操作相同行。但在决赛中为了便于实现 TLB 移除了该设计。

使用参数化配置 tag、index 宽度，参数化配置行大小及路数。当前 icache 和 dcache 配置均为 2 路 8KB，每行 32 字节。

cache-AXI 转换桥内置 store buffer，在 cache 写入 AXI 时非阻塞，可配置写入队列深度。

2.5 TLB 设计

TLB 使用了两级设计。L1 TLB 是两个大小为一项的缓存，用于加速查找操作，用时一拍；L2 TLB 有较大的容量 32 项，查找用时四拍，读、写、INVTLB 操作当拍完成。

查找 TLB 时，输入的变化被认为是一次查询，TLB 将即刻停止当前的查询进度，并在耗时一拍复位后重新启动查找。当在对应查找端口的 L1 缓存中命中或在四拍后 L2 得出结果时，对应查找端口的 ok 信号置 1。为维护缓存之间的一致性，在进行写操作或 INVTLB 操作时，将清空 L1，在下次查找操作时重填；写和 INVTLB 操作直接对 L2 进行。

每个查找端口维护一个两位的状态，分别是 CACHE、LOOKUP、FETCH 和 REFILL，对应 L1 查找和读取、L2 查找、L2 结果读取，以及 L1 的重填。信号变化取消当前查找时，需要一拍返回初始状态，而一拍后输入寄存器更新，已经检测不到输入变化，因此设置一组 cancel_flag 记录取消情况并添加到转移条件中，以此实现取消查找。

2.6 CSR 寄存器

2.7 例外

MeowCPU 实现了下表列出的例外。手册中的 `ADEM` 例外由于手册未给出具体的触发条件没有实现。`FPD` 和 `FPE` 由于不支持浮点指令集未实现。`IPE` 由于没有对特权指令进行权限检查也未实现。

对于中断，MeowCPU 将其标记在下一条从指令队列取出的指令上，并作为例外进行处理。例外返回指令 `ERTN` 在处理器内部也被实现为一种例外。

Ecode	例外代号	例外类型
0x0	INT	中断
0x1	PIL	load 操作页无效例外
0x2	PIS	store 操作页无效例外
0x3	PIF	取指操作页无效例外

Ecode	例外代号	例外类型
0x4	PME	页修改例外
0x7	PPI	页特权等级不合规例外
0x8	ADEF	取指地址错例外
0x9	ALE	访存指令地址错例外
0xB	SYS	系统调用例外
0xC	BRK	断点例外
0xD	INE	指令不存在例外
0x3F	TLBR	TLB 重填例外

2.8 分支预测设计

分支预测单元用于在取指阶段直接预测指令的跳转情况，包括方向预测和目标预测两部分。本模块首先使用 BTB 表，根据 PC 值预测该指令的类型（直接跳转、间接跳转、CALL 或 RET）和直接跳转、间接跳转的目标，同时经由局部历史表，使用 RAS 预测 RET 指令的目标、使用 PHT 预测条件跳转指令的方向。最终经由 BTB 的指令类型预测结果，选择对应的目标预测结果并输出。

各模块的规格：

1. BTB 为 128 项两路组相联，每项包含 PC 值、指令类型和目标地址，将由 PC 得到的 8 位哈希值作为 tag 进行查询。
2. BHT 大小为 64 项，将由 PC 得到的 6 位哈希值作为 tag 进行查询，分支历史长度为 4。
3. PHT 大小为 128 项，将分支历史与 PC 进行哈希作为 tag 进行查询。
4. RAS 为 16 项。

3 成果展示

我们重新编译了 Linux ，将 cache line 大小设置为 32B，并加入了 MeowCPU 的启动画面。最终成功地通过 PMON 将自己编译的 Linux 加载到 Flash 上，并在 MeowCPU 上启动了 Linux 操作系统。

由于时间不足，我们没有成功完成 soc 的设计。故系统展示使用了 chiplab 的示例 soc，能够使用串口、网口外设，并将轻量级 web 服务器 lighttpd 移植到了 la32r 平台，成功搭建了一个 web 服务器，可向主机提供静态网页。

4 参考资料

- 龙芯架构 32 位精简版参考手册
- 使用了 Vivado 的 IP 核：Multiplier、Divider Generator、Block Memory Generator
- Chiplab (https://gitee.com/loongson-edu/chiplab)

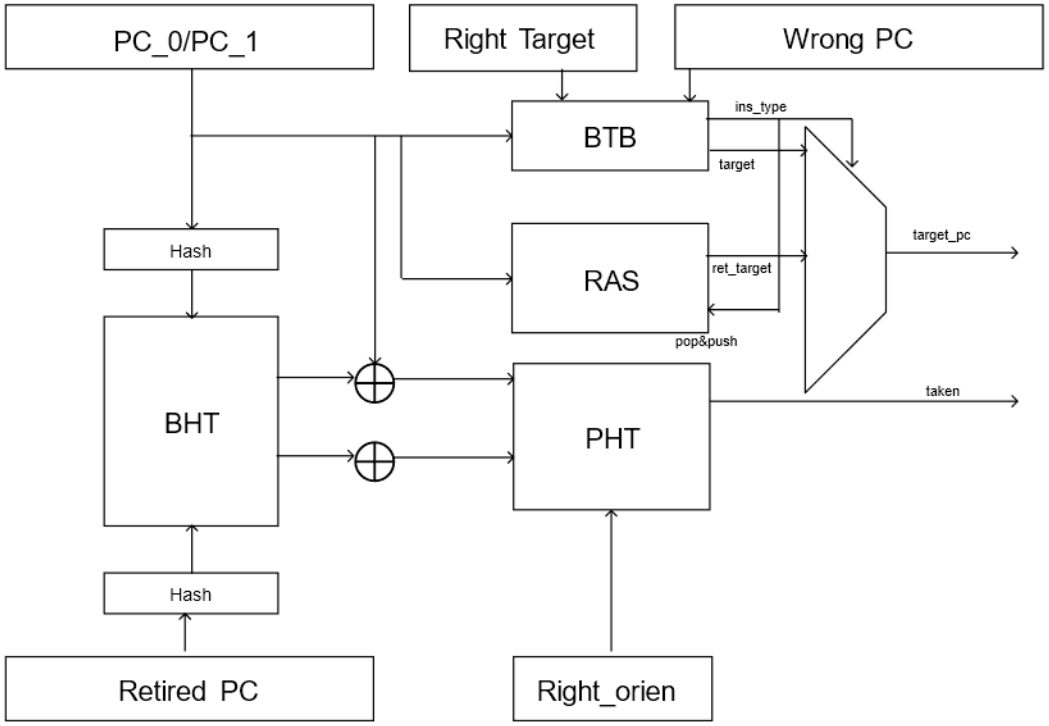

图 2: 分支预测模块框图

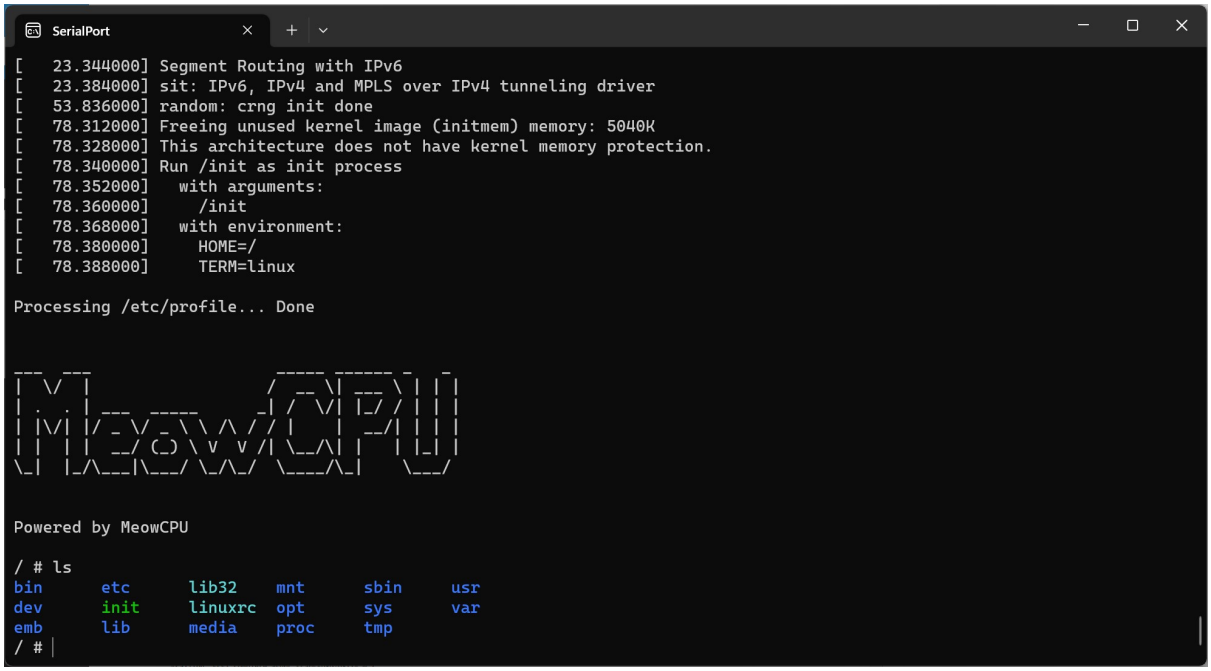

图 3: 启动 Linux

- LA32r_sa （https://gitee.com/MJ_Wang/spinal-loong-arch-core）
- CPU 设计实战. 汪文祥/邢金璋
- 超标量处理器设计. 姚永斌