软工计原联合项目

开发文档

NonExist 组
张钰晖，杨一滨，周正平

目录

Chapter 1

文档说明

本文档是 NonExist 组软工计原联合项目的开发文档。

当初次接到这个项目时，面对卷帙浩繁的文档，我们陷入了一种深深的迷茫，本文档的目的便是描述我们是如何一步一步理清需求，明确分工，最终造出一台 32 位计算机的。

项目从学期第 2 周开始，持续到第 16 周结束，共计 15 周。

查看 GitLab 的 Milestone 功能，我们可以看到每周的 MileStone 如下：

1. **Week 5**: Build CPU baseline
2. **Week 6**: Add instructions
3. **Week 7**: Complete all instructions, build ucore
4. **Week 8**: Complete exception, learn external devices, read books
5. **Week 9**: TLB/MMU/External Devices Required
6. **Week 10**: Link MMU, external devices
7. **Week 11**: Run function test on chips and debug
8. **Week 12**: Run monitor on chip and debug
9. **Week 13**: Run monitor on chip and ucore if possible
10. **Week 14**: Run ucore on chip
11. **Week 15**: More external devices and unit test
12. **Week 16 later**: Document Required

通过观察 Milestone，我们将整个项目周期划分为 5 个 Sprint，每个 Sprint 三星期，其主线可以如下概括：

1. **Sprint 1**：Week 2 - Week 4 查阅相关文献

2. **Sprint 2**：Week 5 - Week 7 指令流水、数据通路

3. **Sprint 3**：Week 8 - Week 10 CP0、异常、MMU、外设连接

4. **Sprint 4**：Week 11 - Week 13 调试功能测例、监控程序

5. **Sprint 5**：Week 14 - Week 16 调试 ucore、撰写文档

当然每个部分还做了一些额外的事情，比如配置开发环境，制作针对 TLB 的功能测例等等，在之后的部分会具体指出。

希望本文档能给读者带来裨益。

Chapter 2

开发周期

我们将开发周期分成 5 个 Sprint，具体展开每个开发周期应该做什么，通过这样详细的拆分，希望读者可以能尽快走出迷茫期，对整个项目有一个大概的认知。

2.1 Sprint 1

Sprint 1 的持续时间为 Week 2 - Week 4，本阶段我们刚刚接触项目，并加上十一国庆节，在这个阶段，我们做了：

1. **阅读文献**: 打印阅读《自己动手写 CPU》和学长文档。《自己动手写 CPU》是一本极其重要的书，作者清晰的思路带领读者一步一步的从零开始搭建出一个流水线框架，因此建议每人一本，快速阅读。

2. **配置环境**: 学习软工平台和 Gitlab，安装 Vivado，配置 Ubuntu 云服务器。由于我们三个人的电脑均不是 Ubuntu，故我们选择购买了一台最低配的云服务器作为我们 Ubuntu 的开发环境，从而实现安装 mips 编译器，编译功能测例、ucore 等等，云服务器给我们提供了完全相同资源共享的环境，且上传下载简单。

3. **明确需求**: 事实上，在这个阶段，尽管我们读了《自己动手写 CPU》和学长文档，但是因为知识面远远不够，我们对项目的认知还是较为模糊，无法进行明确分工。

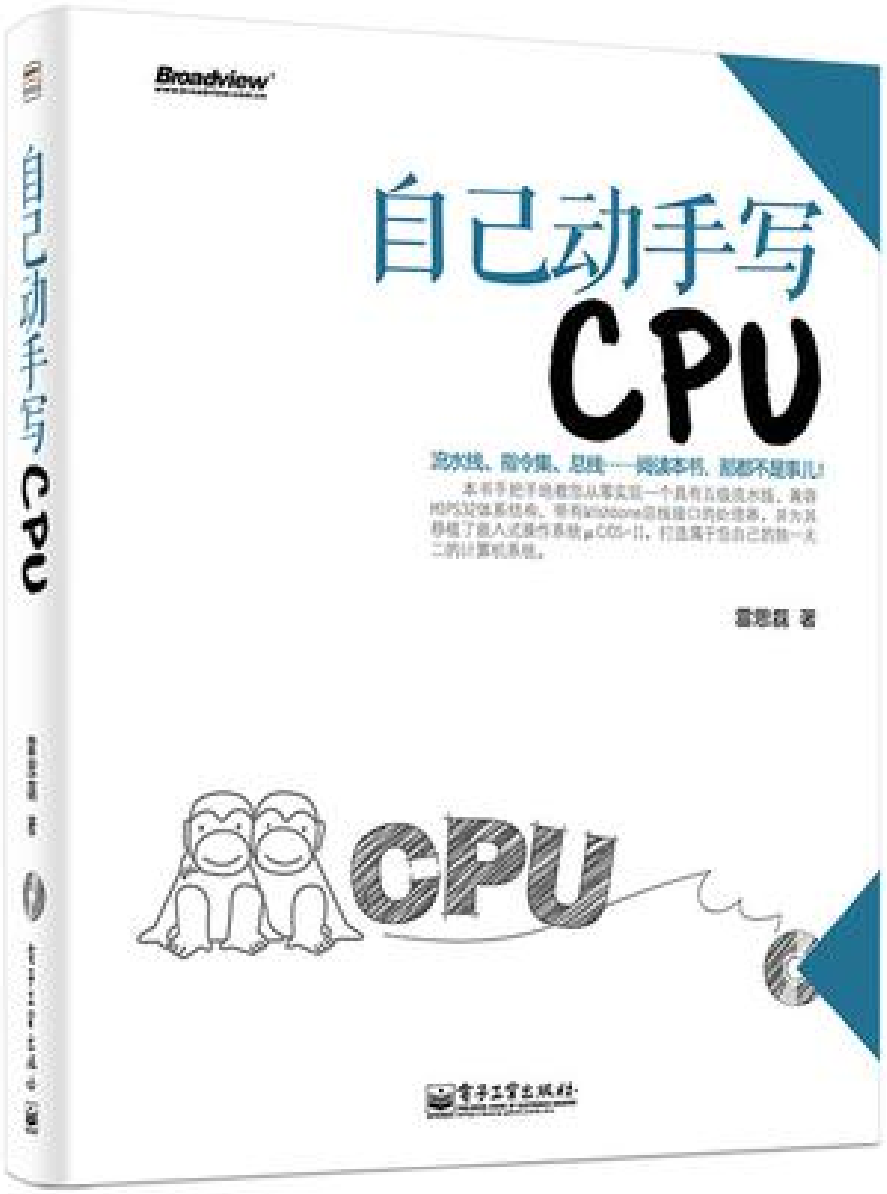

图 2.1: 神书《自己动手写 CPU》

从 Milestone 也可以看出，这段时期我们并没有明显的开发痕迹，主要是阅读资料。

2.2 Sprint 2

Sprint 2 的持续时间为 Week 5 - Week 7。在这个阶段，我们做了：

1. **搭建流水线框架**：实现 MIPS 标准五级流水线，只支持 ori 指令。
2. **增加各类指令**：增加逻辑运算指令、算数运算指令、分支跳转指令、访存指令等 ucore 运行所需的大部分指令。

3. **设计数据通路**：解决流水线竞争与冒险，数据前推，增加控制器实现流水线暂停功能。

4. **单一指令测试**：搭建 SOPC，对实现的每条指令进行简单的测试，检查实现是否正确。

这个阶段是后面所有阶段的基础，推荐的方法是快速阅读与实现《自己动手写 CPU》相关内容，本阶段可以分工，每个成员实现一章的指令，但是每个成员都要读相关章节，这样才能保证所有人能对系统有一个大概的认知。

2.3 Sprint 3

Sprint 3 的持续时间为 Week 8 - Week 10。在这个阶段，我们做了：

1. **异常处理**：实现 CP0 协处理器与异常处理，对其进行相应的的测试。

2. **内存管理**：阅读 TLB、MMU 相关文献，理解 TLB、MMU 的基本原理，实现 TLB、MMU 从而进行内存管理。

3. **外设连接**：了解各种外设（Flash、RAM、串口）的使用方法，写代码对其进行简单测试

4. **仿真操统**：在搭建的云服务器上，编译 ucore，并使用 qemu 进行仿真，了解我们的终极目标是什么。

这个阶段已经算是进阶了，已经对 CPU 基本的框架有了相应的了解，本阶段可以分工，异常处理由一个人完成，推荐的方法是快速阅读与实现《自己动手写 CPU》相关内容，之后《自己动手写 CPU》这本书便完成了它的使命，后期已经没有可以参考直接使用的东西了，最多可以参考该书了解一些相关概念！内存管理推荐一个人完成，上网查阅理解 TLB、MMU 的基本概念，或者向学长老师要一份往年的计原很靠后的 PPT，从而实现内存管理模块。外设连接推荐一个人完成，对提供的顶层代码写一些验证性代码，理解外设的使用方法，掌握外设的时序关系，能阅读相应的英文外设文档。分工后组员一定要进行相互讲解，review 对方的代码，进一步加深对系统的理解。

2.4 Sprint 4

Sprint 4 的持续时间为 Week 11 - Week 13。在这个阶段，我们做了：

1. **调试功能测例**：功能测例内含九十余条指令的测试代码，编译后烧录至 RAM 中便可以测试 CPU 相关指令是否实现正确，功能测例对指令实现要求高，对外设实现要求低。我们先仿真通过了所有功能测例，又使用硬件通过了所有功能测例。

2. **调试监控程序**：监控程序对指令实现要求低，对外设实现要求高，调试需要实现 ROM、Flash 和串口，实现 boot 过程。

进入这个阶段已经没有什么文档可以参考了，因为每个人的问题都不一样，推荐的方法是根据调试文档中的调试技巧，定位问题所在，查阅学长文档和网上的资料，三人共同讨论分析问题、解决问题。

2.5 Sprint 5

Sprint 4 的持续时间为 Week 14 - Week 16。在这个阶段，我们做了：

1. **TLB 功能测例**：依据功能测例的基本框架，对 TLBWR 和 TLBWI 两条指令撰写功能测例。
2. **调试 ucore**：ucore 对指令实现要求高，对外设实现要求高，但由于功能测例和监控程序都已进行了充分的测试，故本阶段调试较为顺利，仅耗时 2 天。
3. **增加外设**：增加显存模块，增加 VGA 输出模块，实现图像输出功能。
4. **文档撰写**：总结经验教训，总结开发心得。

如果前面每个组员都完成了自己的任务，对系统有一个清晰的理解，每项功能都经历了严格的测试，积累了大量的经验，本阶段将不再那么痛苦，但是如果之前阶段得过且过，本阶段可能会遇到无数的问题，因此之前的阶段一定要认真开发，不要划水！

Chapter 3

经验总结

在开发过程中，我们证明了以下方式确实非常值得借鉴：

1. **统一环境**: 我们的 Windows 和 Vivado 版本号全部一致，Ubuntu 采用云服务器，高度一致的环境使得我们不必因为环境的问题而产生莫名其妙的 bug，其他组成员便存在因为编译器版本号等问题导致了结果不同，浪费了大量时间。
2. **共同学习**: 所有人都阅读了相关文档与《自己动手写 CPU》大部分章节，对系统都有一个明确而清晰的认知。
3. **沟通交流**: 每一部分完成后，一定要结合着代码，给其他成员讲清楚你在做什么，怎么实现的这个逻辑，让所有人对系统都有所了解。
4. **及时测试**: 完成每一个功能后，一定要及时测试，即便是最简单的单条指令仿真测试，也能发现很多的问题，一定不要最后堆在一起进行测试，那样会造成成指数增长的调试时间。正是因为我们进行了严格的测试，因此最后调试时间大大缩减。
5. **遵从开发规范**：每周更新 Milestone，每一个任务都单独建立相应的 Git Issue，建立对应分支，完成后发起合并请求，所有人 review 代码，遵从开发规范能保证版本控制得当，责任具体，划分明确，更可为今后文档撰写提供足够的参考。

当然，我们在开发过程中也有一些不足，例如有时候因为别的任务而不得不改变进度，例如第 12 周由于因为作业过多我们本该调试通过的监控程序没有调过，但是这也正是软工“变”的意义所在，拥抱变化，及时调整策略，及时沟通，共同努力，明确任务。

Chapter 4

结语

写到这里，所有的任务也都全部完成，所有的文档也都已经全部写完，一个学期忽然已逝。

正如《自己动手写 CPU》中所说的，冰冻三尺非一日之寒，严谨而细致的开发环节帮助我们及时完成了所有的任务，我们对 CPU 和计算机的理解有了质的提升，真正奋斗一学期，动手实现了一个 20 世纪“简陋”的 CPU，运行上了极其“精简”的 ucore 操作系统

认识的过程是曲折螺旋的，或许作为读者，你现在还完全不能理解这份文档，这并不要紧，你现在只需要对整个系统和工程有一个大致的认知，在不断地阅读与实践中，一点一点的撰写代码、查阅资料，重新阅读相应的章节，相信不远的将来你就能对整个系统有一个很明确的了解，甚至可以指着这份文档说，作者实现的并不好，我有一个更好的实现方法。

最后致敬神书《自己动手写 CPU》，感谢助教和老师一个学期的帮助与指导！