## 当前主流的嵌入式微处理器指令集架构有哪些？请简述每种架构的特点，并调研分析各种架构处理器的国产化情况。

主流微处理器指令集架构：

1. ARM指令集：

所有ARM指令均为32位定长；

使用了“加载-存储(Load-Store)架构” ；

提供了功能强大的一次“加载-存储”多个寄存器的指令；

移位操作可内嵌其他指令中；

所有ARM指令均可条件执行；

体积小、低功耗、低成本、高性能，非常适用于移动通讯领域；

大多数数据操作都在寄存器中完成，指令执行速度更快；

寻址方式灵活简单，执行效率高；

指令长度固定，可通过多流水线方式提高处理效率

1. MIPS指令集

支持64bit指令和操作

一种简洁、优化、具有高度扩展性的RISC架构

有专门的除法器，可以执行除法指令

内核寄存器比ARM多一倍，同样的性能下MIPS的功耗比ARM更低，同样功耗下性能比ARM更高

指令比ARM稍微多一点，稍微灵活一点

采用顺序单发射

包含大量的寄存器、指令数和字符、可视的管道延时时隙，因此MIPS架构能够提供最高的每平方毫米性能和当今SoC设计中最低的能耗

尽量利用软件办法避免流水线中的数据相关问题。

简洁、优化、具有高度扩展性

1. PowerPC

32 个（32 位或 64 位）GPR（通用寄存器）以及 PC、LR（链接寄存器）、CR（条件寄存器）等各种其它寄存器。部分上有 32 个 64 位 FPR（浮点寄存器）。

所有 PowerPC（包括 64 位实现）都使用定长的 32 位指令。

PowerPC处理模型要从内存检索数据、在寄存器中对它进行操作，然后将它存储回内存。几乎没有指令（除了装入和存储）是直接操作内存的。

可伸缩性好、方便灵活，应用范围非常广泛

既有通用处理器，又有微控制器和内核

条款太苛刻，而且比MIPS或ARM的许可证贵得多。

由于开放不足、授权费高昂

1. ARC指令集

ARC处理器IP以追求功耗效率比（DMIPS/mW）和面积效率比（DMIPS/mm2）最优化为目标，出色的硬件微架构使得ARC处理器的各项指标均令人印象深刻。

具有高度可配置性，可通过增加或删除功能模块，满足不同的应用需求，通过不同配置属性实现快速系统集成。

1. RISC-V指令集

指令只有六种格式，并且所有的指令都是 32 位长

RISC-V 指令提供三个寄存器操作数

对于所有指令，要读写的寄存器的标识符总是在同一位置，解码指令之前，即可访问寄存器。

这些格式的立即数字段总是符号扩展，符号位总是在指令中最高位。这意味着可能成为关键路径的立即数符号扩展，可以在指令解码之前进行

一个基于精简指令集（RISC）原则的开源指令集架构（ISA）。

可以自由地用于任何目的，允许任何人设计、制造和销售RISC-V芯片和软件。

设计考虑了小型、快速、低功耗的现实情况来实做，但并没有对特定的微架构做过度的设计。

各种架构处理器国产化情况：

1. ARM指令集处理器

ARM架构设计授权模式分为三个等级：  
使用层级授权：

用户只能购买已经封装好的 ARM 处理器核心，而如果想要实现更多功能和特性，则只能通过增加封装之外的 DSP 核心的形式来实现。大多数缺乏研发设计能力的初创企业都选择购买这种授权；

内核层级授权：

指可以一个内核为基础然后在加上自己的外设形成 MCU，例如三星、德州仪器(TI)、博通、飞思卡尔、富士通等等  
架构/指令集层级授权：

可以对 ARM 架构进行大幅度改造，甚至可以对 ARM 指令集进行扩展或缩减，例如苹果的 A 系列、高通的骁龙系列、华为的麒麟系列。

其中我国的华为麒麟系列在收到制裁前属于深层次的架构/指令集层级授权，根据自家坚持芯片自研深耕多年，积累经验，重金研发的经验和技术，同时为中国首屈一指拥有ARM深层授权的企业，华为的麒麟处理器在移动端的高性能通用处理器上颇具竞争力，发展形势相当好。ARM官宣v9架构不受美国出口管理条例（EAR）的约束，拥有v8架构永久授权的华为可以在第一时间使用该技术，相信在解决芯片制造瓶颈后华为麒麟芯片可以继续发挥荣光，研发高端芯片。

飞腾公司是中国国防科技大学高性能处理器研究团队建立的企业。

2016年公布的FT2000定位于高性能服务器、行业业务主机等。兼容ARMv8指令集，使用自研内核。性能方面包括64个FTC661处理器核，Spec 2006测试成绩与Intel的Xeon E5-2699v相当。

1. MIPS指令集处理器

MIPS的架构授权，并不限制任何对MIPS架构的更改，是一个完全开放的架构，可以在此丰富生态软件基础上进行大规模的创新修改，故对于我国的芯片研发有很大的帮助。我国使用MIPS指令集架构的芯片如下：

龙芯最初是国家投资的中科院计算机所下属课题组，获得了中科院重大知识创新工程项目和国家863计划的支持，通过了严格的成果鉴定、基准程序测试和产品测试，可进入商品化生产；还得到了各地政府和企业的大力支持。龙芯是采用授权+自主指令集的模式进行研发的，基于MIPS深度开发，设计退出了多款龙芯芯片并受到市场认可，2020年，龙芯中科基于二十年的CPU研制和生态建设积累推出了龙芯架构（LoongArch），包括基础架构部分和向量指令、虚拟化、二进制翻译等扩展部分，近2000条指令，龙芯架构具有完全自主、技术先进、兼容生态三方面特点。

龙芯架构从整个架构的顶层规划，到各部分的功能定义，再到细节上每条指令的编码、名称、含义，在架构上进行自主重新设计，具有充分的自主性。

龙芯架构摒弃了传统指令系统中部分不适应当前软硬件设计技术发展趋势的陈旧内容，吸纳了近年来指令系统设计领域诸多先进的技术发展成果。同原有兼容指令系统相比，不仅在硬件方面更易于高性能低功耗设计，而且在软件方面更易于编译优化和操作系统、虚拟机的开发。

龙芯CPU已用于包括北斗在内的十几种国家重器中。

2021年4月，龙芯发布新一代自主指令系统架构——龙芯架构（LoongArch），放弃了以往的MIPS授权，拥有2500多条自主指令，还可以翻译MIPS、ARM及x86指令。

龙芯的目标是在2025年的时候消除指令集之间的壁垒，彻底搞定不同指令集的兼容问题。

计划组建LoongArch联盟，一方面免费开放LoongArch，另一方面也要在高校推广，取代RISC-V。

龙芯架构在设计时充分考虑兼容生态需求，融合了各国际主流指令系统的主要功能特性，同时依托龙芯团队在二进制翻译方面十余年的技术积累创新，不仅能够确保现有龙芯电脑上应用二进制的无损迁移，而且能够实现多种国际主流指令系统的高效二进制翻译。

除了龙芯这样的大型科研项目外,一家名为君正的民营芯片公司也成为了MIPS的客户,这也是MIPS在中国市场的第二个客户。与龙芯不同，君正是一家彻彻底底面向商业化的企业型公司，其生产的MIPS芯片主要面向MP3，MP4和智能手机。除了这两家正规企业外。还有众多的山寨厂商也在使用MIPS的芯片。

1. RISC-V 指令集处理器

阿里平头哥：玄铁 910（XuanTie910）支持16核，主频 2.5GHz，单核性能达到 7.1 Coremark/MHz。两大创新在于：采用3发射8执行的复杂乱序执行架构，是业界首个实现每周期 2 条内存访问的 RISC-V 处理器；基于 RISC-V 扩展了 50 余条指令，系统性增强了 RISC-V 的计算、存储和多核等方面能力。

可用于设计制造高性能芯片，应用于5G、人工智能以及自动驾驶等领域。

嘉楠耘智、比特大陆：基于SiFive开源核的AI芯片；

华米：基于SiFive核的边缘AI计算芯片黄山一号量产；

紫光展锐: 采用RISC-V技术的春藤系列蓝牙耳机芯片；

艾派克: 采用RISC-V的打印机管理芯片；

优微科技: 基于RISC-V的快冲芯片。

## 调研ARM指令集架构的发展历程，简述每一代架构的特点，并重点分析ARMv8和ARMv9架构的特点和应用前景

ARMv1：基本数据处理（无乘法）；

字节,字以及多字 load/store；

软件中断；

包括子程序调用指令BL；

26 bit 地址总线 （64M）；

处于实验的ARM体系结构版本，不具有实用性；

ARMv2：Multiply & Multiply-accumulate，MUL乘法指令引入；

支持协处理器；

有了集成内存缓存；

在FIQ模式提供多个影子寄存器；

支持线程同步；

提供原子性的load-and-store——SWP和SWPB；

集成了MEMC(MMC)、图形、I/O处理器；

26 bit 地址总线；

ARMv3：寻址空间增至32位（4GB）；

当前程序状态信息从原来的R15寄存器移到当前程序状态寄存器CPSR中（Current Program Status Register）;

增加了程序状态保存寄存器SPSR（SavedProgram Status Register）；

增加了两种异常模式，使操作系统代码可方便地使用数据访问中止异常、指令预取中止异常和未定义指令异常；

增加了MRS/MSR指令，以访问新增的CPSR/SPSR寄存器；

增加了从异常处理返回的指令功能。变种中有无协处理器总线的版本

ARMv4：首次使用五级流水线，变种中有使用三级流水线；

支持半字读取和写入；

完善了软件中断SWI指令的功能；

添加带符号的字节和半字数据指令；

把一些未使用的指令空间捕获为未定义指令；

ARMv4T中引入了16位的Thumb指令集（T变种）；

添加了特权模式，提高系统权限隔离，防止越权非法操作，提高安全性；

此代开始不兼容26位寻址方式，26位寻址方式被弃用；

此代为目前支持的最老的ARM架构；

ARMv5：增强型DSP指令集，包括全部算法操作和16位乘法操作；

提供字节代码执行的硬件和优化软件加速功能；

带有链接和交换的转移BLX指令；计数前导零CLZ指令；BRK中断指令；

增加了数字信号处理指令（V5TE版）； 为协处理器增加更多可选择的指令

改进了ARM/Thumb状态之间的切换效率；

支持新的JAVA，提供字节代码执行的硬件和优化软件加速功能。

ARMv6：增加了存储系统异常处理，多核环境；

改进了上下文切换逻辑和说明；

SIMD通过添加并行加法和减法指令，改善提高了多媒体声音/视频处理能力；

支持正式的内存模型，包括 1 级缓存支持以及对对齐和字节序支持的修订；

添加TrustZone 技术：让硬件支持两组地址空间，非安全区的程序不能访问安全区，提供帐户安全，密匙管理功能；

THUMBTM：35%代码压缩；

DSP扩充：高性能定点DSP功能；

JazelleTM：Java性能优化，可提高8倍；

ARMv7：采用了强大的信号处理扩展集；

产品线根据目标应用场景侧重分为“A” “R” “M”三类

A Profile: 应用于复杂的虚拟内存OS；

R Profile：实时控制处理器；

M Profile：低功耗应用；

划为CortexTM Family；

引入了架构配置文件的概念；

所有ARMv7均实现了Thumb-2技术；

所使用NEON技术的增进提高了DSP和多媒体处理吞吐量约400%；

支持多种可选拓展：

安全扩展；多处理扩展；大型物理地址扩展；

虚拟化扩展；通用计时器扩展；性能监视器扩展

ARMv8：ARM公司首款支持64位指令集的处理器架构；

向下兼容AArch32 32位指令集；

首次引入4级异常分级设计，其中 EL0 用于应用程序，EL1 用于操作系统，EL2 用于虚拟化，EL3 用于安全固件

核心数量可以从1个到4个不等；

集成了NEON SIMD引擎、ARM CoreSight多核心调试与追踪模块、128-bit AMBA ACE一致性总线界面，还可选加密加速单元，能将加密软件的运行速度提升最多10倍。

ARMv9：‎高级 SIMD、可扩展矢量扩展指令集（SVE、SVE2）

可扩展矩阵扩展指令集（SME）‎；

内存标签扩展（MTE）可让开发者找出软件中时间、空间安全性违规；

适用于机器学习；

Arm机密计算体系结构（Arm Confidential Compute Architecture：CAA）

Realm Management Extension（RME）：增强机密安全功能；

BRBE：提供分析信息，如自动FDO；

嵌入式跟踪扩展（ETE）和跟踪缓冲区扩展（TRBE）增强了ARMv9的跟踪功能；

TME：对 Arm 架构的硬件事务内存支持；

现ARMv8和ARMv9均多用于通用高性能移动处理器中，其中ARMv9在兼容ARMv8的基础上，提升处理器性能，同时提升安全性、增强矢量计算、机器学习及数字信号处理的问题处理能力；此外，ARMv9处理器将不再局限于移动/嵌入式市场，现已经扩展到PC、HPC高性能计算、深度学习等新市场。

ARMv9与性能有关的重要升级就是SVE2指令集，SVE2可以支持128为倍数、最多2048位运算因此SVE2可以增强ML机器学习、DSP信号处理能力，提升未来5G、虚拟现实、增强现实以及CPU本地运行ML的性能，同时ARM未来还会继续提升AI人工智能性能。在于数据安全方面上，全新的CCA机密计算体系架构基于TrustZone安全技术，并引入动态域技术。对操作系统及管理程序来说是完全不透明的，不会被系统或者软件提权攻击，而且依然可以接受管理及调度。ARMv9将成为未来10年内数以千亿级arm芯片的基础。

## RISC-V架构的设计哲学是什么，请简介RISC-V架构的典型组成和特点，并以蜂鸟E200为例阐述一个典型RISC-V架构处理器的设计内容有哪些。

**RISC-V设计哲学：**

“大道至简”，力图通过架构的定义使硬件的实现足够简单。

1. **架构的篇幅小，轻量**：RISC-V架构具备后发优势，可以有效规避计算机体系结构发展过程中暴露出的各种问题，提高性能。且不存在向后兼容的问题。
2. **模块化的指令集，拓展性极强**：模块化的RISC-V架构能够使得用户能够灵活选择不同的模块组合，以满足不同的应用场景，共同的部分则可以相互兼容。
3. **指令数量精简，易于上手，保留核心**：短小精悍的架构以及模块化的哲学，使得RISC-V架构的指令数目非常的简洁。基本的RISC-V指令数目仅有40多条，加上其他的模块化扩展指令总共几十条指令。

**典型组成和特点：**

1. **模块化指令子集**

RISC-V的指令集使用模块化的方式进行组织，每一个模块使用一个英文字母来表示。RISC-V最基本也是唯一强制要求实现的指令集部分是由I字母表示的基本整数指令子集。使用该指令子集，便能够实现完整的软件编译器。其他的指令子集部分均为可选的模块。

1. **可配置的通用寄存器组**

RISC-V架构支持32位或者64位架构，RISC-V架构的整数通用寄存器组包含32个（I架构）或16个（E架构）通用寄存器，其中整数寄存器0被预留为常数0，其他的31个或15个位普通的通用整数寄存器。

1. **规整的指令编码**

得益于后发优势和总结了多年处理器发展的经验，RISC-V的指令集编码非常规整，指令集所需的通用寄存器的索引（Index）都被放在固定的位置，因此指令译码器可以非常便捷地译码出寄存器索引，然后读取寄存器组。

1. **简洁的存储器访问指令**

使用专用的存储器读（Load）和存储器写（Store）访问存储器（Memory），其他普通指令无法访问存储器。存储器访问的基本单位是字节。存储器读写指令支持一个字节（8位）、半字（16位）、单字（32位），如果是64位架构还支持双字（64位）。

支持地址对齐和非对齐的架构，处理器可以选择通过硬件或软件来支持。仅支持小端存储格式。不支持地址自增自减。采用松散的存储器模型，对于访问不同地址的存储器读写指令的执行顺序不做要求。

1. **高效的分支跳转指令**

RISC-V架构有两条无条件跳转指令JAL和JALR，6条带条件跳转指令，该指令与普通的运算指令一样直接使用两个整数操作数，然后对其进行比较。如果比较的条件满足，则进行跳转。因此此类指令将比较和跳转两个操作放在一条指令里完成

1. **简洁的子程序调用**

放弃使用多寄存器指令，大幅简化CPU的硬件设计。在需要“保存现场”和“恢复现场”的场合，可以使用公用的程序库来进行。

1. **无条件码执行**

RISC-V放弃使用这种带“条件码”指令的方式，对于任何的条件判断都使用普通的带条件分支跳转指令

1. **无分支延迟槽**

RISC-V放弃使用分支延迟槽，因为现代高性能处理器的硬件动态分支预测算法精度已经非常高，可以有强大的分支预测电路保证CPU能够精准地预测跳转执行。

1. **无零开销硬件循环**

零开销硬件循环大幅增加了硬件设计的复杂度，这与RISC-V架构简化硬件的设计哲学完全相反，因此也没有采用。

1. **简洁的运算指令**

RISC-V架构使用模块化的方式组织不同的指令子集。

最基本的整数指令子集（I）支持的运算包括加法、减法、移位、按位逻辑操作和比较操作，并可通过组合或函数的方式完成更多的复杂操作，整数乘除法指令子集（M）支持有符号或者无符号的乘法和除法操作；单精度浮点指令子集（F）与双精度浮点指令子集（D）支持浮点加减法、乘除法、乘累加、开平方根和比较等操作，同时提供整数与浮点、单精度与双精度浮点之间的格式转换操作。

任何运算指令错误均不产生异常，而是产生某个特殊的默认值，同时设置某些状态寄存器的状态位。

1. **优雅的压缩指令子集**

RISC-V具有后发优势，从一开始编规划了压缩指令，预留了足够的编码空间，16位指令与32位指令可以无缝自由地交织，处理器也没有定义额外的状态。

16位指令的压缩策略是将一部分普通最常用的32位指令中的信息进行压缩重排得到，因此每一条16位长的指令都能找到其一一对应的原始32位指令。RV32C的代码体积比RV32减少40％，并且与ARM、MIPS和x86架构相比有不错的表现。

**以蜂鸟E200为例阐述一个典型RISC-V架构处理器的设计内容**

蜂鸟E200是基于RISC-V指令集架构设计的低性能要求CPU，其设计理念与RISC-V的设计哲学有高度重合之处。其设计理念如下：模块化和可重用性、面积最小化、结构简单化、性能不追求极端。

1. **流水线**

严格来讲，蜂鸟E200是一个变长流水线结构，也可以非严谨地定义为二级流水线。

流水线的第一级为“取指”，由IFU完成。

流水线的第二级为“译码”、“执行”、“写回”，前两者由EXU完成，第三个由WB完成，这三项操作均处于同一时钟周期执行。

流水线的第三级为“访存”，由LSU完成，但是LSU写回的结果仍然需要通过第二级的WB模块写回通用寄存器组，故并不是严格的三级流水线。

1. **取指实现**

蜂鸟E200的IFU微架构主要包括以下功能：简单的分支预测，根据PC的地址访问ITCM或BIU，对取回的指令进行简单译码，生成取指的PC。

为提高取指速度并优化连续取指，蜂鸟使用ITCM进行指令的存储以满足实时性的要求，直接将取回的指令在同一个周期进行部分译码，如果为分支/跳转指令，则IFU直接在同一周期内进行分支预测，并使用译码和分支预测得出的信息生成下一条待取指令的PC。此操作在同一周期内完成了指令读取、部分译码、分支预测和下一条指令PC生成等步骤。

蜂鸟E200处理器面向超低功耗的嵌入式应用，在性能和复杂度方面做了一定的取舍。为了低功耗、小面积的目的，舍弃了很多复杂的技术。如只采用静态分支预测，仅对ITCM访问进行优化，而对BIU访问放弃优化。保证常见的情形下性能可观。如推ITCM区间内的顺序取指，不管地址是否对齐，都能做到“快”和“连续不断”。

1. **执行实现**

蜂鸟E200系列CPU是两级流水线架构，其译码、执行、交付、写回功能全部处于流水线的第二级，由执行单元EXU完成，具体功能如下：将IFU通过IR寄存器发送给EXU的指令进行译码与派遣；通过译码得出的操作数寄存器索引读取寄存器组；维护指令的数据相关性；将指令派遣给不同的运算单元执行；交付指令；将指令的运算结果写回寄存器组。

此过程涉及译码、执行写回功能，译码过程中其中涉及证书通用寄存器组、CSR寄存器的设计；指令发射派遣中要考虑到数据冲突的解决方式；ALU设计中包括多周期乘除法器、访存地址生成、普通ALU运算、多分支预测解析、CSR读写控制的设计，这5个功能子单元共享一份运算数据通路；此外浮点单元也是影响运算精度的重要部分。

由于蜂鸟E200是一种两级流水线的微架构，其“执行”阶段的EXU单元事实上包含了经典五级流水线种对应的“译码”“执行”“写回”功能，还包含了“交付”功能。因此EXU单元是蜂鸟E200处理器核的心脏

1. **交付硬件实现**

蜂鸟E200处理器将“交付”安排在“执行”阶段。只要前序的指令没有发生“分支预测错误”“中断”或者“异常”，就可以判定此条指令能够被成功“交付”。需要注意的是，对于分支预测错误的分支指令自身和遭遇了中断或者异常的指令自身而言，仍然属于成功“交付”的指令，因为她们自身已经被真正执行且对处理器的状态真正地产生了影响。

在此阶段中，蜂鸟延续了RISC-V架构处理器对于交付简化的特点。表现于指令没有条件码；不需要处理单条指令“取消”的情形；所有的运算指令都不会产生异常；只需要处理分支预测指令错误预测造成的后续指令流取消、中断和异常造成的后续指令流取消这两类流水线冲刷情形。

1. **写回实现**

蜂鸟E200的写回策略是一种因地制宜的混合策略：指令分为单周期指令和长指令两大类；长指令的“交付”和“写回”分开，使得即便执行了多周期长指令，仍然不会阻塞流水线，让后续的单周期指令仍然能够顺利地写回和交付。

在此运行过程中，最终写回仲裁、长指令写回仲裁、OITF（Outstanding Instruction Track FIFO）滞外指令跟踪是需要设计的重要组件。

综合来说，蜂鸟E200处理器的写回是一种混合策略：如果仅讨论单周期指令，其策略属于“顺序发射，顺序执行，顺序写回”；如果仅讨论长周期指令，其策略属于“顺序发射，乱序执行，乱序写回”；如果单周期指令和多周期指令统一考虑，则属于“顺序发射，乱序执行，乱序写回”。

1. **储存器架构实现**

蜂鸟E200随RISC-V一样对存储器访问指令做了简化，包括仅支持小端模式、无地址自增自减模式、无“一次读多个数据”和“一次写多个数据”指令。

存储器子系统包括为读写指令以及“A”扩展指令生成存储器访问地址的AGU，存储器访问的控制模块，用作指令存储部件，也可用于存储数据而被读写指令访问ITCM，

数据存储部件DTCM：。

蜂鸟E200处理器有专用的总线分别访问ITCM和DTCM；ITCM也有一组输入ICB总线接口来自LSU的访问，即ITCM所在的地址空间也可以通过LSU被读和写指令访问，从而可以用于存储数据读和写指令对于ITCM的访问主要用于程序的上电初始化。在程序正常运行时，不推荐将数据段放入ITCM。ITCM的主体是一块64位的单口SRAM，其大小和及地址可以通过Config.v中的宏定义参数配置。

1. **自定义总线协议ICB**

蜂鸟E200处理器定义了一种自定义总线协议ICB（Internal Chip Bus），用于处理器核内部使用，也可以作为SoC中的总线。

对于蜂鸟E200这样的低功耗处理器，ICB总线能够被用于几乎所有的场合，包括作为内部模块之间的接口、SRAM模块接口、低速设备总线以及系统存储总线等。

1. **总线接口单元BIU**

BIU主要负责接收来自IFU和LSU单元的存储器访问请求，并使用标准的ICB接口，然后通过判断其访问的地址区间来访问外部的不同接口，接口包括：快速IO接口、私有外设接口、系统存储接口和CLINT接口、PLIC接口。此过程中需要BIU微架构和SoC总线微架构的设计。并未提到蜂鸟E200在此方面有特殊设计差异。

1. **中断和异常**

在中断屏蔽上，蜂鸟E200只实现了机器模式，监督模式和用户模式下的中断使能位（SEIE、UEIE等）无任何意义；在中断等待上，也只实现了机器模式，其他模式对应的中断等待状态位无任何意义。蜂鸟E200是“只支持机器模式”架构，且没有实现MPU与MMU（不会产生虚拟地址相关的异常），因此只支持RISC-V架构中机器模式相关的异常。此外，蜂鸟E200只实现了RISC-V架构定义的3种基本中断类型，并未实现更多的自定义中断类型。

在中断接口上，蜂鸟E200在处理器顶层接口中有4根中断输入信号，分别是软件中断、计时器中断、外部中断和调试中断。SoC层面的CLINT模块产生一根软件中断信号和一根计时器中断信号，通给蜂鸟E200处理器核；SoC层面的PLIC接入多个外部中断源将其仲裁后生成一根外部中断信号，通给蜂鸟E200处理器核。SoC层面的调试模块生成一根调试中断，通给蜂鸟E200处理器核。所有的中断信号均由蜂鸟E200处理器核的交付模块进行处理。

对于交付模块对终端和异常的处理，交付模块接受来自CLINT和PLIC的3根中断信号的请求，蜂鸟E200的实现中将中断作为一种精确异步异常，这种异常的“异常返回地址”将会为下一条尚未交付的指令。当异步异常和ALU造成的同步异常以及中断同时发生时，优先级依次为：长指令造成的异步异常优先级最高，中断造成的异步异常其次，ALU造成的同步异常最后。异常一旦发生，便会冲刷流水线，将后续的指令取消掉，并向IFU模块发送冲刷请求和重新取指指令的PC，用以重新从新的PC地址开始去指令。

1. **调试机制**

蜂鸟E200不仅开源了处理器核的实现、SoC实现、FPGA平台和软件示例，还实现和开源的调试方案，具备完整的GDB交互调试功能。常用的两种是“交互式调试”和“追踪式调试”。其中涉及DTM模块、硬件调试模块、调试中断处理、调试机制CSR寄存器多方面的设计。

蜂鸟E200处理器和调试机制的硬件严格依据“0.11版本”定义的方案，目前仅支持交互式调试。其硬件实现如下：调试主机为PC端的调试平台，软件的开发编译在主机PC端完成，并且在主机PC端运行调试软件；主机PC端的GDB软件与其Gdbserver通信，Gdbserver可以用开源软件OpenOCD充当，该软件包含了各种常见的硬件芯片驱动，如USB转JTAG。芯片USB接口通过USB连接线与主机PC连接，芯片的JTAG接口则可以与RISC-V处理器的SoC硬件平台连接。在RISC-V的SoC中，JTAG接口由DTM模块转换成内部的调试总线，通过该总线访问调试模块。

## 结合蜂鸟E200处理器的设计过程和tinyCPU的实验经验，自己设计一款RISC-V处理器。给出该处理器的设计理念、指令集设计方案、功能模块组成、总线结构等，并尝试在Vivado环境下对部分关键模块进行设计与仿真验证。（可参考实验中tinyCPU的设计流程，但要体现出自己的特色）

**设计理念**

本cpu是以实验的tinyCPU为思考起点，以改进tinyCPU不足之处为目标，设计的一款采用Load-Save结构、低功耗、\_\_\_\_\_的软核处理器，在此过程中参考了RISC-V指令集架构中处理问题的思路以及曾使用过的MIPS指令集架构。

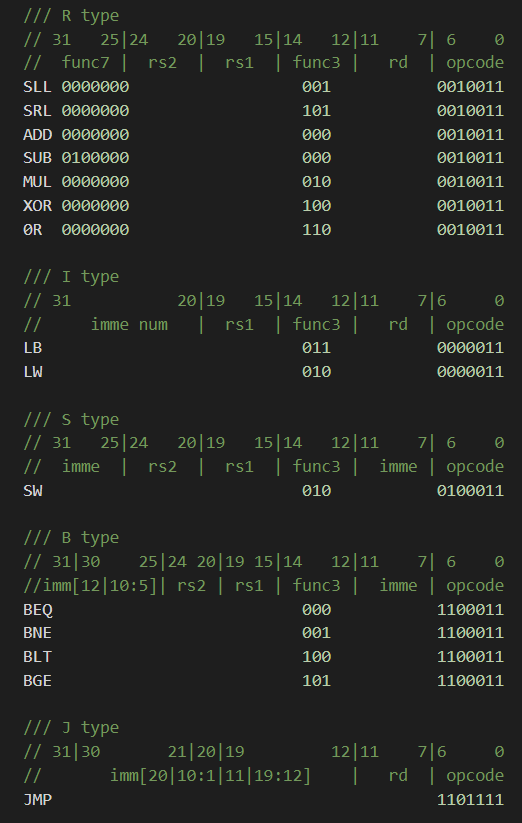
实验中所设计的Tinycpu没有通用寄存器，使得程序难以脱离RAM完成较为复杂的处理。Tinycpu中为了简化设计使用了堆栈结构作为存储结构，数据只得从栈顶存取，极大程度上限制了栈中数据的访问；且alu的运算过程会消耗栈内数据，效果等效于从栈顶弹出计算所用个数的数据并将结果压回栈，对于稍复杂的程序(如值交换、判断跳转)的编写会比较繁冗，需要多次读写内存或在栈中多次压入数据的方式解决。前者会降低程序运行速度，后者会浪费资源且增加程序编写的困难，所以我认为使用通用寄存器堆代替堆栈能有效解决上述问题(尽管会使结构较为复杂)。

此外，tinycpu中使用的指令流程是状态机，简化了设计面积，同时避免了模块相关性问题。但使得处理器执行效率和硬件利用率过低，不太能满足现今对低功耗嵌入式设备处理性能的需求，所以我认为使用多级流水线代替状态机是有一定的必要。而考虑到功耗、处理器面积以及个人设计水平有限的因素，我选择与Cortex-M0相似的二级流水线。就算是最低限度的二级流水线，也是存在同一指令周期完成多条指令处理的情况，多种不同的数据需要同时从不同的地方向不同的目标传输。当然，流水线的设计就需要模块之间耦合度较低，复用性变差，每一个模块都需要特殊设计，这会导致较大的设计开销。

至于异常和中断的设计，会使结构复杂度大幅上升，鉴于我理解浅显、为了设计结构的简单，本cpu设计中不考虑此模块设计。

**指令集设计方案**

这里参考了RISC-V原版的32位指令集的格式，构建出定长32位指令集，并分成由于实现有限，这里只列出部分指令，其中包括15条指令，按照类型分成R型、I型、S型、L型和J型，虽然不很完善，但各类均有涉及，其余未实现的指令实现方法也类似。指令具体格式如下：



使用此种指令集编码设计的好处便是能根据对指令某一位的判断即可缩小确定操作类型，同类的操作会被归为一类以减少分支过多带来的硬件复杂程度增加。

**功能模块组成**

（1）取指模块

取指阶段的功能是产生新的pc值，并从存储器中读取相应位置的指令。正常情况下，会从0取值开始运行，每完成一次操作pc值会自增+4(每条指令均是对齐32位的)确定下一条指令的偏移位置，遇到跳转指令，pc寄存器会从总线中读取偏移量，CPU再从pc所指地址取指。为了实现简单的二级流水线，需要考虑到运行到跳转指令时的流水线冲刷问题，故pc的输入中有跳转信号的输入。设计的pc寄存器模块的输入输出信号如下所示：

clk 输入 1 时钟输入信号

rst 输入 1 复位输入信号

load 输入 1 跳转标志

d 输入 32 跳转地址，即跳转到该地址的偏移量

inc 输入 1 PC地址自增

q 输出 32 PC寄存器值，即从该值处取指

IR寄存器从数据线读取指令并预处理访存指令和跳转指令，让较慢的访存指令提前开始准备数据供后续操作使用，ir寄存器模块的输入输出信号如下所示：

clk 输入 1 时钟输入信号

rst 输入 1 复位输入信号

load 输入 1 跳转标志

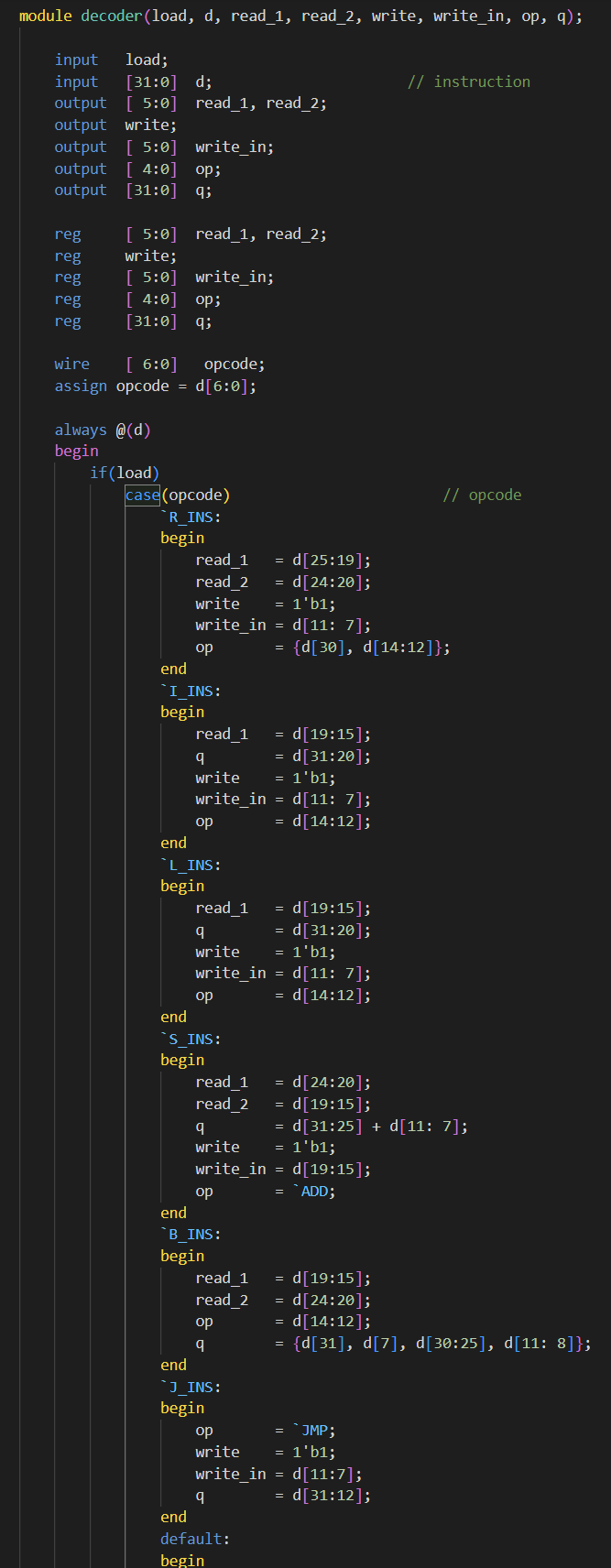
d 输入 32 跳转地址，即跳转到该地址

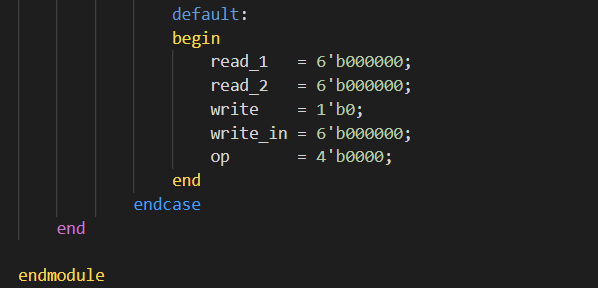
q 输出 32 IR寄存器存储的指令，即下一条需要执行的指令。当需要访

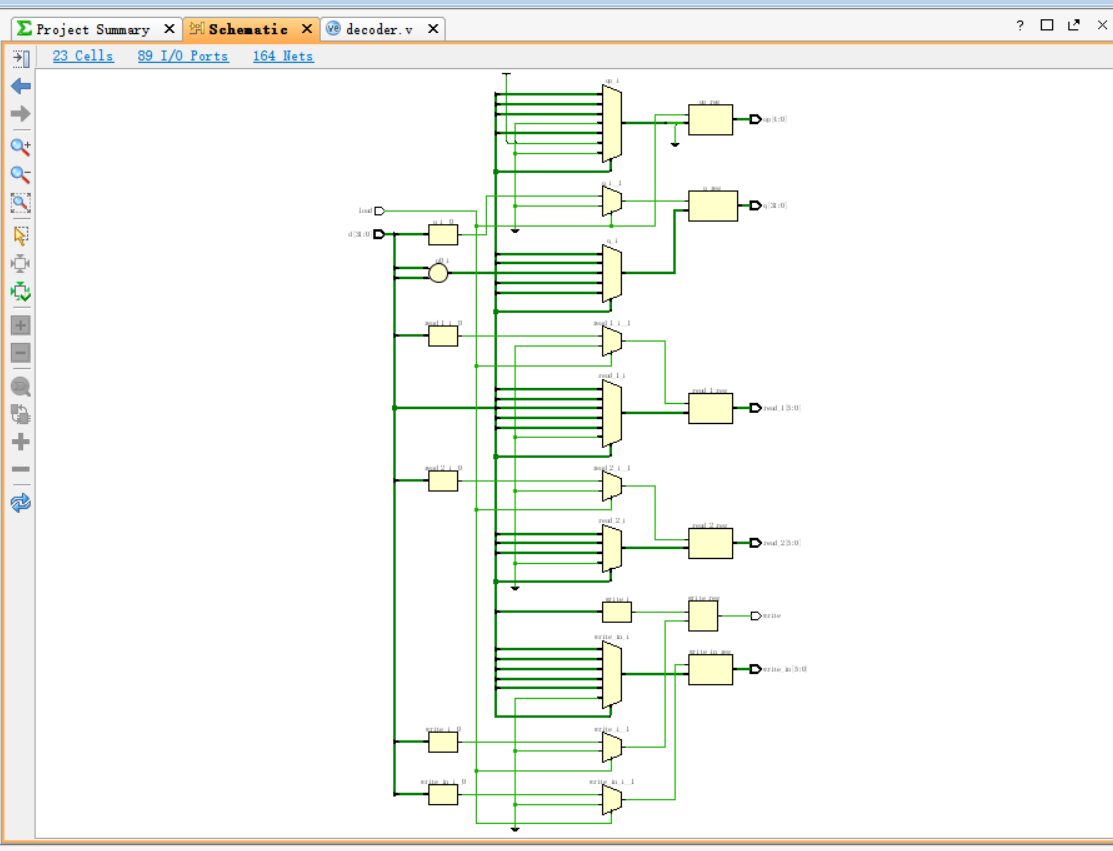
存或跳转时，IR通过此输出涉及的地址让RAM寻址输出。

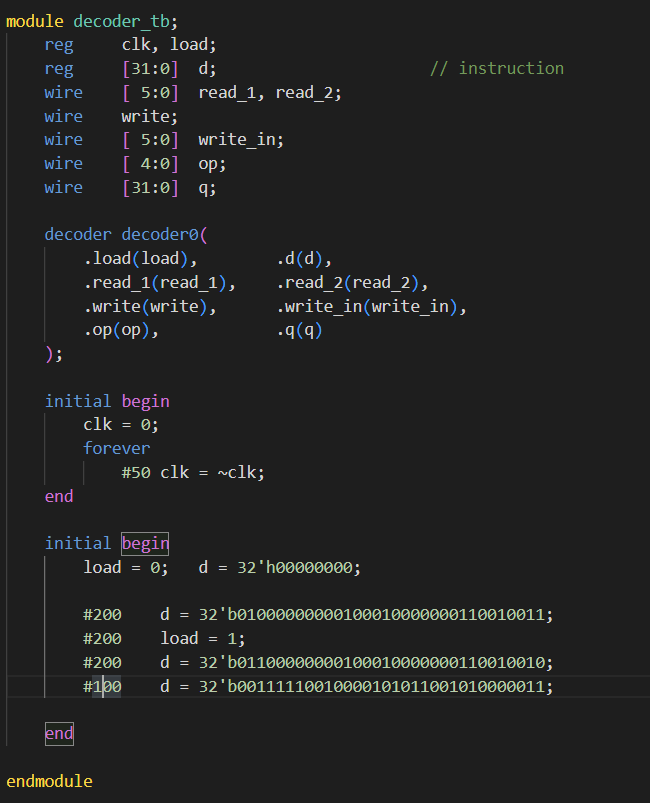
（2）译码模块

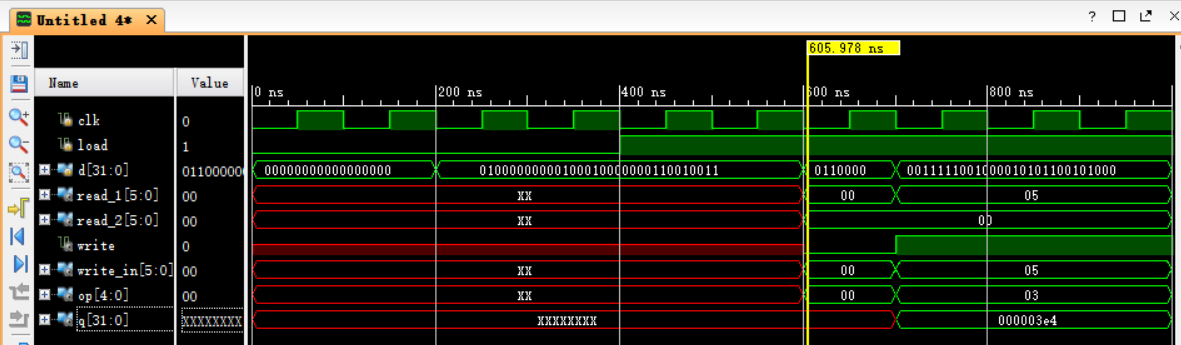
译码模块主要负责以下的作用：根据指令内容，解析出当前具体是哪一条指令；根据具体的指令，确定当前指令涉及的寄存器，是读寄存器操作还是写寄存器操作，读寄存器是一个还是两个，写寄存器写哪一个寄存器此类；访问通用寄存器，让寄存器准备好要读的寄存器的值。诸如此些的细节操作可通过指令集中指令位的设置同样考虑到流水线的实现，跳转冲刷也需要受到统一控制，设计的译码模块的输入输出信号如下所示：





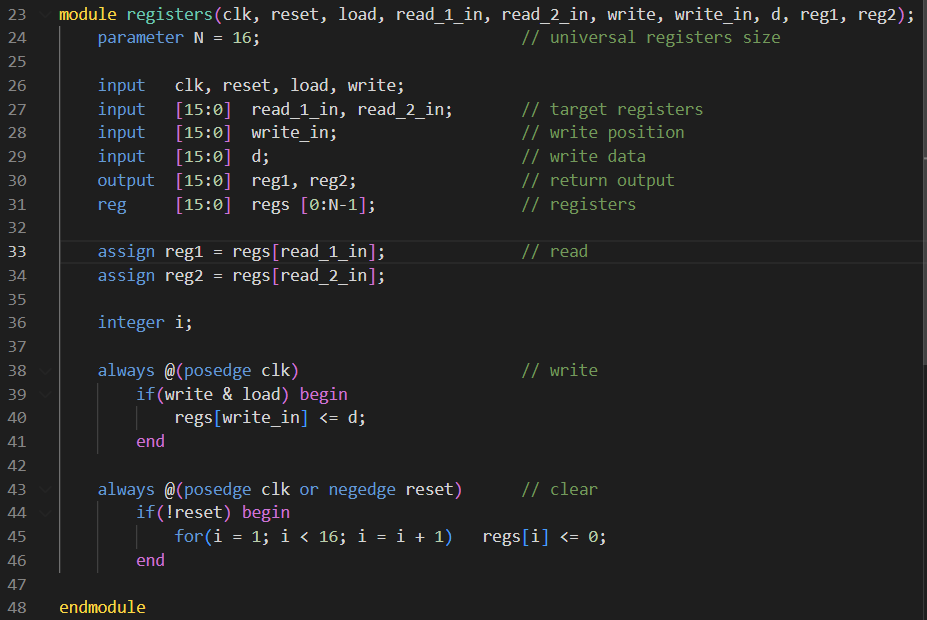


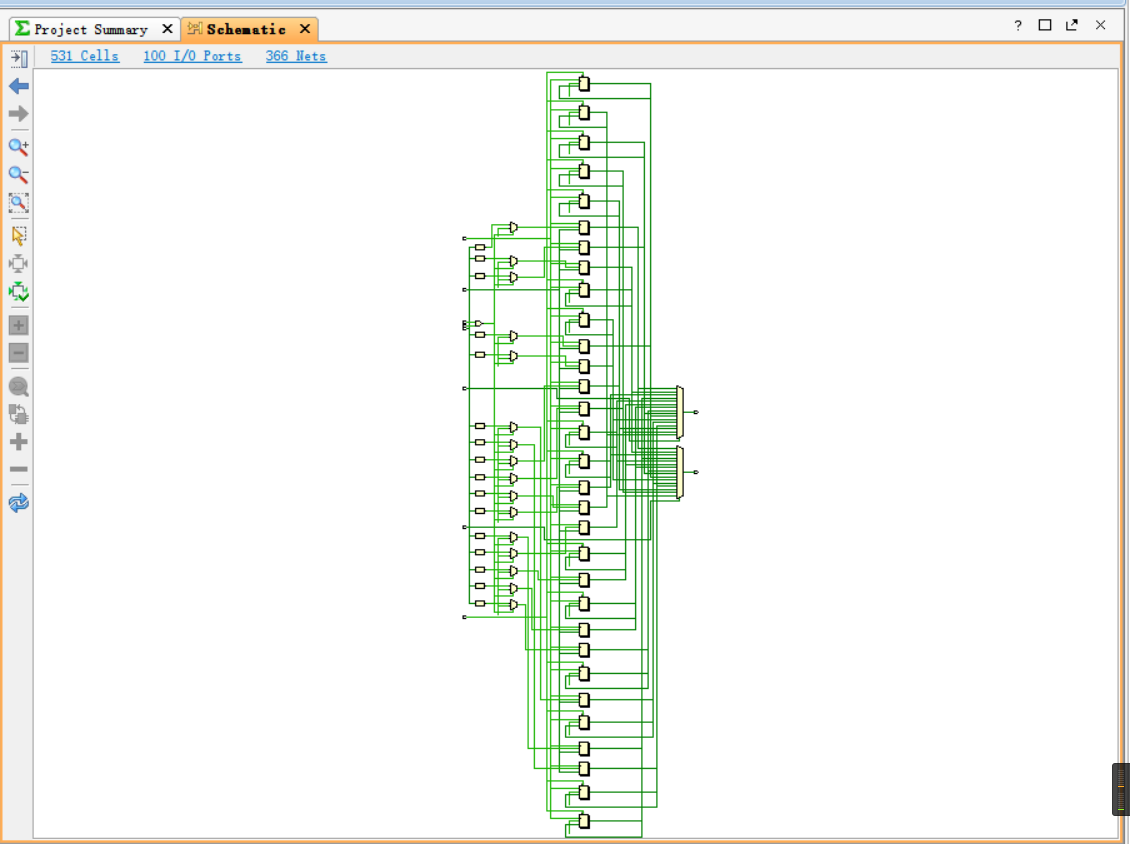


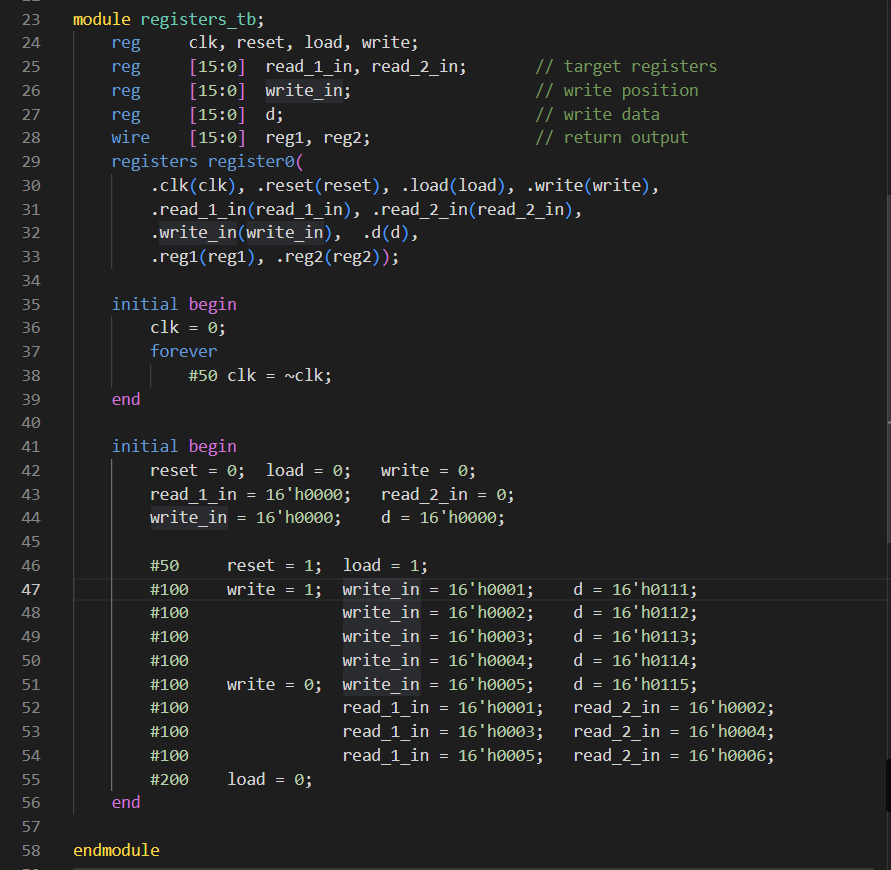


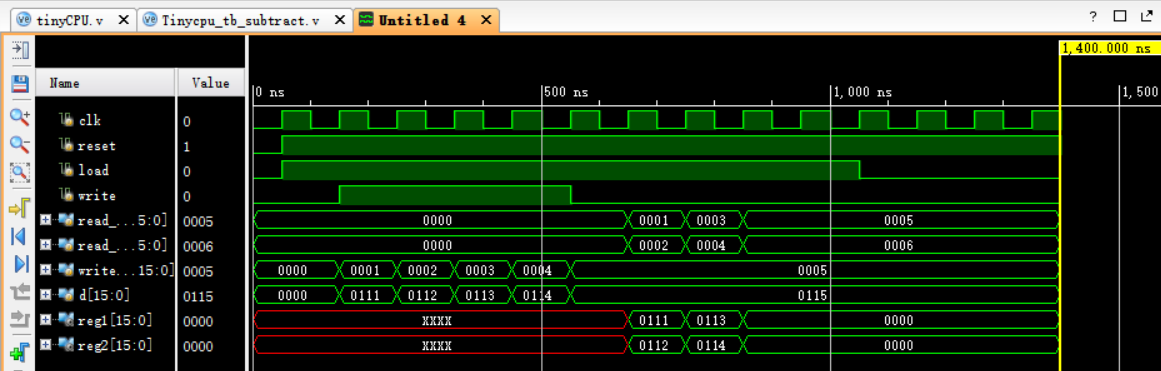
（3）寄存器堆

为解决复杂程序逻辑过程中数值存放多堆栈结构不灵活依赖RAM内存读写速度慢的问题设置的，具体定义如下：









（4）执行模块

alu 与实验中的tinyCPU结构功能类似，负责数字与逻辑运算，支持一元运算和二元运算，计算方式由译码器模块决定，计算所用数据由寄存器组或译码器提供，输出的计算结果经过加工传回译码器完成向寄存器堆的写回。若信号为跳转信号，则协助前文所涉及到的IR寄存器一同完成指令跳转的过程。Alu执行计算模块的输入输出信号如下所示：

a 输入 32 操作数1，一元运算时取此输入作为操作数

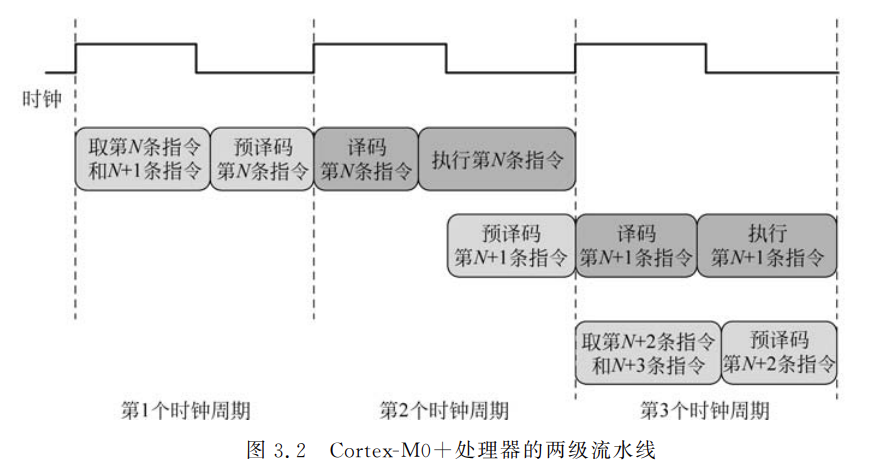
b 输入 32 操作数2，i型指令则数据来源于译码器，或源于寄存器堆

f 输入 5 计算方式

s 输出 32 计算结果

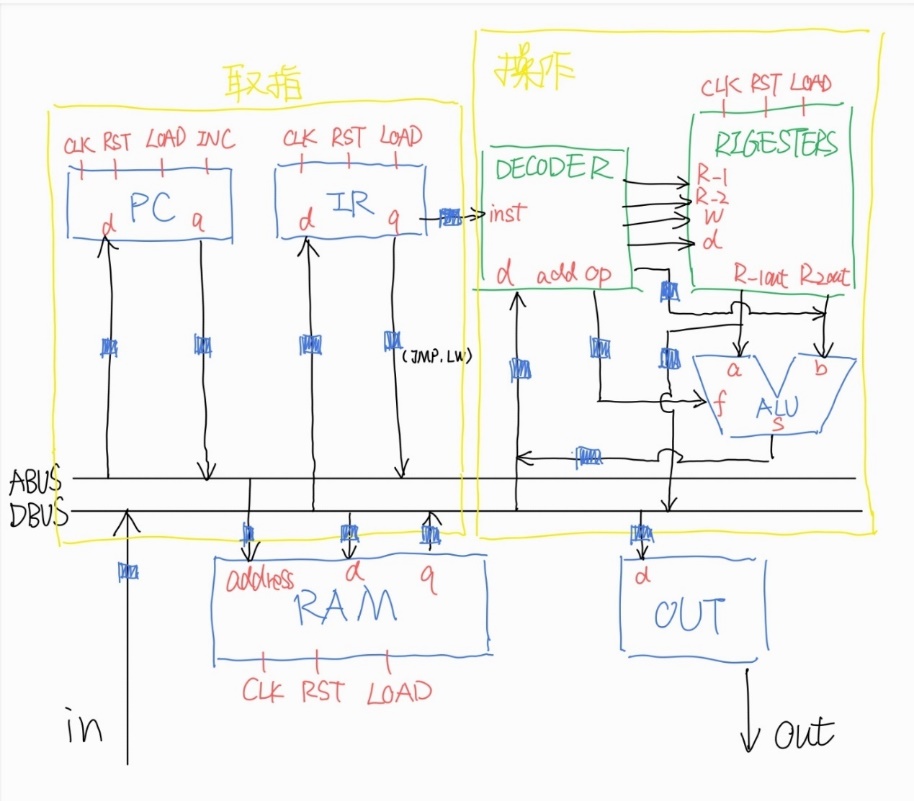
（5）流水线控制

二级流水线的设计参考了Cortex-M0的两级流水线的实现，第一级流水线完成取指和预译码，第二级流水线完成主译码和执行，如下如所示，这与后面总线结构中划分的两个模块相对应。Cortex-M0为减少取值过程中访存次数，选择一次性取两条相邻的指令，本cpu简化设计，舍弃了访存次数减少的优势，仍保留为一周期一次取指。当出现跳转指令时，需要冲刷流水线，弃掉在jmp指令在执行过程中取指到和预译码的指令，更新pc指针并从新的位置开始取指，虽然这过程会导致一个时钟周期的延迟，但跳转指令在程序中占比并不是主要地位，整体下来相比于状态机模式还是有较大的运行效率提升。



**总线结构**

总线结构如下图所示，虽然最佳情况是内部和外部组件的通信剥离开为最佳，但精力、能力、时间所限智能凑出这么一个耦合度较高，有较高隐患和错误可能的总线结构。途中蓝色方框表示沿袭了实验中所使用到的元件并加以小小修改；绿色方框表示较于实验有修改新添加的组件；在肩头上的蓝色方块表示该通路的连接受到调控，红色文字标识端口名简写；黄色框标出了两级流水线操作涉及的两个主要模块。



综上，是我综合课上所学、实验所想、资料查阅后以我拙见完成的一款简陋的处理器。感谢您一学期的教导和悉心解惑，通过一学期的课程学习，我在以蜂鸟E200处理器为例开展的教学中深入了解了处理器的基本结构、各组件间协作逻辑、模块设计中的常见问题及解决方案等等。进一步加深了对计算机体系结构的认识理解，辛苦您了！