



**2024~2025学年 第一学期**

**《计算机体系结构》**

大作业

班 级\_\_\_\_\_计算机科学与技术22-4班\_\_\_\_\_\_\_\_\_

学 号 \_ \_\_2022217477\_\_\_\_\_\_\_\_\_\_

姓 名\_\_\_\_\_ 侯腾跃\_\_\_\_\_\_\_\_\_

2024年11月

# intel CPU结构与技术

Intel 4004

Intel 4004是全球第一款商用单芯片微处理器，，最初是为日本Busicom公司设计的，用于其计算器产品。后来，英特尔公司在除计算器芯片市场之外的其他市场上自由出售4004芯片，这标志着英特尔从单一的存储器制造商向微处理器制造商的转型。以下是它的一些硬件组成细节：

1. 尺寸和封装：Intel 4004的尺寸为3cm×4cm，外层有16只针脚，采用16-pin CerDIP（陶瓷双列直插式封装）。

2. 晶体管数量：4004微处理器片内集成了2,300个MOS（金属-氧化物-半导体）晶体管。

3. 制程技术：它采用10微米（10,000纳米）PMOS（正型金属氧化物半导体）制造技术，电路线宽为10微米。

4. 总线宽度：拥有4位的内部数据总线和4位的外部数据总线，以及12位地址总线。

5. 运算能力：能执行4位运算，支持8位指令集，每秒可以处理高达92,000个命令。

6. 生产：最初的4004微处理器采用金顶白色陶瓷封装，后期出现了普通黑陶瓷以及塑料封装（P4004）。

7. 系统组成：Intel推出的MCS-4微型计算机系统包括4001 ROM芯片、4002 RAM芯片、4003 移位寄存器芯片和4004微处理器。这四款芯片共同构成了一个微型计算机系统。

Intel 8080

Intel 8080是英特尔公司于1974年4月1日发布的一款8位微处理器。是早期8位微处理器中的一个重要里程碑，它的设计和性能在当时被认为是非常先进的。它被广泛应用于各种控制系统和嵌入式系统中，并且对后来的微处理器设计产生了深远的影响，特别是对Intel 8086 CPU架构的影响，后者催生了x86系列处理器 。此外，Intel 8080还推动了个人电脑革命，它被用作第一台个人电脑Altair 8800的处理器，从而开启了个人电脑发展的新纪元 。以下是它的一些关键硬件组成：

1. 晶体管数量：集成了6,000个晶体管 。

2. 制造工艺：采用了6.0 µm NMOS技术 。

3. 内部和外部数据总线宽度：均为8位 。

4. 地址总线宽度：16位，支持高达64 KB的内存 。

5. 寄存器：包含7个8位寄存器，其中6个可以组合成3个16位寄存器 。

6. 封装形式：采用40-pin陶瓷DIP-40封装 。

7. 输入输出端口：拥有256个I/O端口，允许I/O设备连接而不需要分配内存空间 。

Intel 8086

Intel 8086是英特尔公司在1978年推出的一款16位微处理器，Intel 8086的推出标志着第三代微处理器的问世，它的设计允许运行更为复杂和灵活的应用程序。8086的架构成为了后续处理器设计的基础，并且它的指令集一直被后续的x86兼容处理器所保留。此外，8086还是第一款包含微码（microcode）的Intel处理器，这使得它能够通过更新微码来修正芯片中的错误或增加新功能 。以下是Intel 8086的硬件组成：

1. 晶体管数量：包含约29,000个晶体管 。

2. 制造工艺：采用3.0 µm HMOS技术 。

3. 内部和外部数据总线宽度：均为16位宽 。

4. 地址总线宽度：20位，支持高达1024 KB（1 MB）的内存寻址 。

5. 封装形式：采用40-pin PDIP（塑料双列直插式封装）。

6. 指令集：包含约300个操作指令 。

7. 运算能力：具备0.8 MIPs的处理能力 。

8. 内存管理：能够直接寻址高达1 MB的内存 。

9. 辅助运算芯片：与之配合的数学辅助运算芯片（Math Co-processor）是8087，用于提升数学运算效率 。

Intel Pentium

Intel Pentium是英特尔公司推出的一系列x86架构的微处理器。以下是Pentium系列中几个重要型号的硬件组成：

1. Pentium处理器：

MMX技术：添加了Intel的MMX指令集，增强了CPU在多媒体应用中的性能。

L1 Cach：升级为双倍大小，即16 KB指令+16 KB数据Cache。

时钟速度：最初推出的Pentium时脉速度为66 MHz和60 MHz，后来逐渐推出了更高频率的版本，如75, 90, 100, 120, 133, 150, 166, 200, 和 233 MHz。

2. Pentium Pro：

核心代号：P6架构。

晶体管数量：约550万个晶体管。

时钟频率：150/166/180和200MHz四种。

缓存：16KB的一级缓存和256KB的二级缓存，二级缓存与CPU同频运行。

封装技术：采用了“PPGA”封装技术，将二级缓存芯片与CPU芯片封装在一起。

3. Pentium II：

核心代号：Klamath和Deschutes。

制程技术：最初使用350纳米制程，后来的Deschutes使用250纳米制程。

时钟频率：从233 MHz到450 MHz不等。

缓存：二级缓存位于CPU芯片外部的模块上，运行在处理器时钟的一半频率。

4. Pentium III：

架构：基于P6微架构，加入了MMX指令集。

制程技术：最初使用0.25微米制程，后来改进为0.18微米。

时钟频率：从450 MHz到1.4 GHz不等。

缓存：二级缓存大小从256 KB到512 KB不等，运行在处理器时钟的一半频率。

5. Pentium 4：

晶体管数量：约4200万个晶体管，Northwood核心集成了5500万个晶体管。

制程技术：最初采用0.18微米制程，后来改进为0.13微米。

时钟频率：从1.5 GHz开始，逐步提高。

缓存：最初为256 KB，后来增加到512 KB，甚至1 MB或2 MB。

前端总线：最初为400 MHz，后来提高到533 MHz、800 MHz和1066 MHz。

这些Pentium系列处理器在不同的时期代表了英特尔在微处理器设计和制造方面的技术进步。随着时间的推移，Pentium系列处理器在性能、功耗、集成度和功能方面都有了显著的提升。

Intel Core 2 Duo

Intel Core 2 Duo是英特尔公司推出的一款双核处理器，它基于全新的Core微架构设计，该处理器的架构设计允许动态共享最后一级高速缓存，提高了缓存的效率。此外，它还引入了增强的安全性、虚拟化、以及集成在处理器内部的可管理性等特点。Core 2 Duo处理器在性能和能效方面相比前代产品有显著提升，提供了卓越的性能和更快的系统响应速度 。具有以下主要硬件组成：

1. 内核数：2个执行内核。

2. 光刻技术：45纳米或65纳米工艺技术。

3. 处理器基本频率：不同型号的处理器频率不同，例如E8400型号的基本频率为3.00 GHz 。

4. 缓存：大多数型号拥有3MB或6MB的L2 Cache，例如E8400型号拥有6MB L2 Cache 。

5. 总线速度：不同型号的处理器总线速度不同，例如E8400型号的总线速度为1333 MHz 。

6. 热设计功耗（TDP）：不同型号的TDP不同，例如E8400型号的TDP为65W 。

7. VID电压范围\*\*：不同型号的电压范围不同，例如E8400型号的VID电压范围为0.8500V-1.3625V 。

8.封装规格：支持的插槽类型，例如E8400型号支持LGA775插槽 。

9.先进技术：包括Intel® 虚拟化技术 (VT-x)、Intel® 64位技术、增强型Intel SpeedStep®技术等 。

Intel Sandy Bridge

Intel Sandy Bridge是英特尔在2011年推出的微处理器架构，它在多个方面进行了重要的改进和创新。以下是Sandy Bridge的一些关键硬件组成和特点：

1. 集成图形核心：Sandy Bridge将CPU和GPU集成在同一芯片上，这是一次革命性的改变。GPU拥有最多12个执行单元，支持DirectX 10.1和OpenGL 2.1，性能显著提升 。

2. 环形总线架构：Sandy Bridge采用了全新的环形总线（Ring Bus）架构，连接了计算内核、图形核心、三级缓存和系统代理。这种设计提高了数据传输效率并降低了延迟 。

3. 高级矢量扩展指令集（AVX）：Sandy Bridge支持AVX指令集，这使得处理器在处理浮点运算时更为高效，尤其是对科学和工程应用 。

4. 增强的分支预测单元：新的分支预测单元提高了预测精度，减少了分支预测失误率，从而降低了功耗并提升了性能 。

5. 物理寄存器文件（PRF）：Sandy Bridge使用了物理寄存器文件，这有助于提高乱序执行的效率和性能 。

6. 三级缓存：Sandy Bridge拥有高速的三级缓存，其设计允许更高的带宽和更低的延迟，从而提高了处理器的性能 。

7. 内存控制器和PCI-Express接口：Sandy Bridge集成了内存控制器和PCI-E控制器，这有助于提高内存访问速度和扩展能力 。

8. 功耗控制：Sandy Bridge优化了功耗控制，允许处理器根据工作负载动态调整CPU和GPU的功耗分配，以实现更高的能效比 。

9. 睿频加速技术（Turbo Boost）：Sandy Bridge引入了第二代睿频加速技术，允许处理器在不超过热设计功耗（TDP）的情况下，动态提高活跃核心的频率，以提升性能 。

10. 超线程技术（Hyper-Threading）：Sandy Bridge支持超线程技术，允许每个核心同时处理两个线程，从而提高了多任务处理能力 。

Intel Alder Lake

Intel Alder Lake是英特尔公司推出的第12代酷睿处理器，采用了一系列创新技术，以下是其主要的硬件组成和特点：

1. 混合架构：Alder Lake采用了性能核（Performance-core，P-core）和能效核（Efficient-core，E-core）的混合架构设计。这种设计将专注于主要工作负载的性能核与专为多任务处理而构建的能效核相结合，以优化性能和功耗。

2. 核心和线程：Alder Lake处理器拥有多达16个内核和24个线程，其中桌面版最高配置为8个性能核和8个能效核。

3. 缓存：Alder Lake拥有高达30MB的英特尔智能高速缓存（L3 Cache）。

4. 图形处理单元（GPU）：集成了基于Xe架构的英特尔超核芯显卡，具有多达96个图形执行单元（EUs），支持HDMI 2.1，可驱动多达四块4K60 HDR显示器或一块8K分辨率显示器。

5. 深度学习加速：支持英特尔深度学习加速（Intel® DL Boost），CPU上集成了VNNI指令，GPU上集成了INT8/dp4指令，以加速AI推理工作负载。

6. 内存支持：支持DDR5-4800和DDR4-3200内存，提供更高的内存速度和容量。

7. PCIe通道：提供多达16个PCIe 4.0通道，以及支持PCIe 5.0的就绪通道，增强了与高速设备如SSD和显卡的连接能力。

8. 接口和扩展性：支持Thunderbolt™ 4或USB 4，提供高速数据传输和视频输出能力。

9. 硬件线程调度器：Alder Lake配备了英特尔硬件线程调度器2，智能地指示操作系统将适当的工作负载分配给合适的内核，优化工作负载在性能核和能效核上的执行。

10. 制程技术：基于Intel 7制程技术（10nm Enhanced SuperFin）制造。

Intel Arrow Lake

Intel Arrow Lake是英特尔即将推出的第15代酷睿处理器，它采用了先进的硬件结构和混合架构设计。以下是Arrow Lake的一些关键硬件组成和特点：

1. 核心架构：性能核心（P-core）基于新的Lion Cove架构，能效核心（E-core）基于新的Skymont架构。支持高达8个性能核心和16个能效核心的组合，共计24个核心。

2. 缓存设计：P-core拥有高达3MB的L2缓存，E-core拥有1.5MB的L2缓存，所有核心共享高达36MB的L3缓存。

3. 制程技术：Compute tile使用台积电N3B工艺，GPU tile采用台积电N5P工艺，SoC tile与I/O tile使用N6工艺。

4. 集成图形：集成了基于Xe-LPG架构的GPU，拥有4个Xe核心（512个着色器），支持DirectX 12 Ultimate和PCIe 5.0。

5. AI加速：集成了NPU 3单元，提供最高13 TOPS的AI性能，加上GPU和CPU的AI性能，总AI性能可达36 TOPS。

6. 内存支持：支持DDR5内存，原生支持6400 MT/s，支持高达192GB的内存容量，每DIMM支持高达48GB的密度。

7. 扩展接口：提供多达20条PCIe 5.0通道和4条PCIe 4.0通道，支持Thunderbolt 4和Wi-Fi 6E。

8. 封装技术：使用LGA-1851新规格插槽，与800系列芯片组主板搭配使用。

Arrow Lake与Lunar Lake

Arrow Lake与Lunar Lake有着相同的新一代CPU内核，包括Lion Cove P-Core和Skymont E-Core，所以它们被划为同一代产品，两者也是由多个模块组成并采用Foveros先进封装技术组合在一起，但除此之外两者其实就没啥相似的地方了。   
  
 Lunar Lake是由计算模块和平台控制模块所组成，计算模块包含CPU、NPU和GPU，内存控制器、媒体引擎、显示引擎、IPU也在里面，基本上就是一个传统单芯片CPU没了PCIe控制器，而平台控制模块则包含PCIe、USB、Thunderbolt 4、WiFi、千兆有线网络MAC，此外还有安全模块，它的功能和传统的PCH没啥差异。Lunar Lake还把两颗LPDDR5封装到CPU的PCB上让笔记本主板设计变得非常紧凑，同时还能提升内存频率并降低40%的功耗。   
  
  
而Arrow Lake的结构则沿袭自Meteor Lake，可以说Arrow Lake才是Meteor Lake真正的继任者，CPU同样由计算模块、SoC模块、图形模块、IO模块这四大模块组成，也都用Foveros封装在基础模块上。此外Arrow Lake上也有一个填料模块，它的作用就把顶盖覆盖范围内没有运算模块的部分填满，因为模块和散热顶盖之间需要有充分接触，否则由于压力不同会对芯片造成损伤。   
 Arrow Lake的计算模块是肯定与Meteor Lake上的不一样的，核心规模比采用相同内核的Lunar Lake大得多，最高能到8P+16E。在Lunar Lake上，E-Core并不挂在环形总线上，与P-Core通信要走NOC总线，也不能使用P-Core的L3缓存，所以英特尔是直接把它们叫作LP E-Core的。而Arrow Lake的P-Core与E-Core都在一个环形总线内，可共享最多36MB L3缓存。   
 Arrow Lake的SoC模块大概就是Meteor Lake上的小改，里面的NPU依然是第三代的，不是Lunar Lake上的第四代NPU，与Meteor Lake的SoC模块相比新增了对PCIe 5.0的支持，至于LP E-Core，目前在今天发布的Arrow Lake-S桌面处理器上是没有的，其他的移动处理器不好说。显示模块使用的是与Meteor Lake相同的Xe-LPG架构，但规模减半，IO模块应该是不一样的，因为桌面平台需要提供完整的PCIe x16，而移动版只有PCIe x8。   
关于Arrow Lake各模块的制程工艺，计算模块用的是台积电N3B，图形模块使用N5P工艺，SoC和IO模块则是N6工艺，除了计算模块制程与Meteor Lake不一样外其他都是一样的。Arrow Lake与Lunar Lake有着相同的新一代CPU内核，包括Lion Cove P-Core和Skymont E-Core，所以它们被划为同一代产品，两者也是由多个模块组成并采用Foveros先进封装技术组合在一起，但除此之外两者其实就没啥相似的地方了。   
  
 Lunar Lake是由计算模块和平台控制模块所组成，计算模块包含CPU、NPU和GPU，内存控制器、媒体引擎、显示引擎、IPU也在里面，基本上就是一个传统单芯片CPU没了PCIe控制器，而平台控制模块则包含PCIe、USB、Thunderbolt 4、WiFi、千兆有线网络MAC，此外还有安全模块，它的功能和传统的PCH没啥差异。Lunar Lake还把两颗LPDDR5封装到CPU的PCB上让笔记本主板设计变得非常紧凑，同时还能提升内存频率并降低40%的功耗。   
  
 而Arrow Lake的结构则沿袭自Meteor Lake，可以说Arrow Lake才是Meteor Lake真正的继任者，CPU同样由计算模块、SoC模块、图形模块、IO模块这四大模块组成，也都用Foveros封装在基础模块上。此外Arrow Lake上也有一个填料模块，它的作用就把顶盖覆盖范围内没有运算模块的部分填满，因为模块和散热顶盖之间需要有充分接触，否则由于压力不同会对芯片造成损伤。   
Arrow Lake的计算模块是肯定与Meteor Lake上的不一样的，核心规模比采用相同内核的Lunar Lake大得多，最高能到8P+16E。在Lunar Lake上，E-Core并不挂在环形总线上，与P-Core通信要走NOC总线，也不能使用P-Core的L3缓存，所以英特尔是直接把它们叫作LP E-Core的。而Arrow Lake的P-Core与E-Core都在一个环形总线内，可共享最多36MB L3缓存。   
 Arrow Lake的SoC模块大概就是Meteor Lake上的小改，里面的NPU依然是第三代的，不是Lunar Lake上的第四代NPU，与Meteor Lake的SoC模块相比新增了对PCIe 5.0的支持，至于LP E-Core，目前在今天发布的Arrow Lake-S桌面处理器上是没有的，其他的移动处理器不好说。显示模块使用的是与Meteor Lake相同的Xe-LPG架构，但规模减半，IO模块应该是不一样的，因为桌面平台需要提供完整的PCIe x16，而移动版只有PCIe x8。   
关于Arrow Lake各模块的制程工艺，计算模块用的是台积电N3B，图形模块使用N5P工艺，SoC和IO模块则是N6工艺，除了计算模块制程与Meteor Lake不一样外其他都是一样的。

最为最新处理器，Intel Arrow Lake与以前的系统（以Intel 14th Gen Core为例）相比，进行了多项重要的改进和创新：

1. 核心架构：

Arrow Lake：采用新的Lion Cove性能核心和Skymont能效核心，以及新的Xe LPG显卡架构。据报道，Arrow Lake的Core Ultra 7 265K在SiSoftware数据库中的性能测试超过了Ryzen 9 9950X，且在多核性能上比Core i7-14700K高出29%。

14th Gen Core：继续使用性能混合架构，优化游戏、内容创作和生产力任务。支持DDR5内存，提供高达5600MT/s的速度，并向后兼容DDR4。

2. 能效比：

Arrow Lake：主打能耗比，据报道，Arrow Lake CPU在多核性能上与Ryzen 9 9950X相当，但在特定功率目标下，Arrow Lake的性能略高于Ryzen 9 9950X（在160W时）。

14th Gen Core：提供了强大的性能和超频能力，但可能在能效比方面不如Arrow Lake。

3. 集成图形：

Arrow Lake：集成了基于Xe-LPG架构的GPU，提供更高的图形性能和AI加速能力。

14th Gen Core：采用Xᵉ架构的英特尔超核芯显卡，支持高达8K HDR和增强媒体支持。

4. AI加速：

Arrow Lake：集成了NPU，提供高达13 TOPS的AI性能。

14th Gen Core：也提供了英特尔深度学习加速技术，但具体TOPS未提及。

5. 超频和内存特性：

Arrow Lake：支持新的超频特性，包括内存超频和内核超频。

14th Gen Core：提供了一套超频工具，包括英特尔Extreme Tuning Utility和英特尔Dynamic Memory Boost，支持内存超频和处理器超频。

6. 平台特性：

Arrow Lake：需要新的LGA 1851插槽，支持DDR5 6400MT/s内存，提供高达192GB的内存容量。

14th Gen Core：支持英特尔700系列芯片组，向后兼容600系列芯片组，支持高达5600MT/s的DDR5内存。

7. 性能：

Arrow Lake：在某些基准测试中，Arrow Lake-S资格样本在多核性能上比Core i9-14900K快14%，在Cinebench R23中快18%。

14th Gen Core：提供了强大的性能，特别是在游戏和内容创作方面。

**总结感悟：**

技术进步的快速性：从Intel 4004到Intel Arrow Lake，CPU技术的发展速度非常快。每一代CPU都在性能、功耗、集成度和功能上有所提升，这体现了摩尔定律的影响，即集成电路上可容纳的晶体管数量大约每两年翻一番。

创新的重要性：CPU的发展史是一个不断创新的过程。从最初的4位处理器到如今的多核、集成GPU、AI加速等，每一项新技术的出现都极大地推动了计算能力的提升。

性能与功耗的平衡：随着技术的发展，CPU不仅要追求更高的性能，还要在功耗和散热方面做出优化。这反映了在追求性能的同时，能效比也成为了衡量CPU性能的重要指标。

集成化趋势：从单一的CPU到集成GPU、内存控制器等，CPU的集成度越来越高，这不仅提升了性能，也简化了系统设计，降低了功耗。

架构的演变：CPU架构的演变，如从PMOS到NMOS，再到CMOS，以及从单核到多核，再到混合架构，每一次架构的变革都带来了性能的飞跃。

市场竞争的推动作用：CPU市场的竞争，尤其是Intel和AMD之间的竞争，推动了技术的快速发展。这种竞争促使两家公司不断推出新的产品，以保持市场领先地位。

软件与硬件的协同发展：CPU的发展不仅仅是硬件的进步，还需要软件的支持。操作系统、编程语言和应用程序都需要与新的硬件特性相适应，以充分发挥硬件的潜力。

# 一．作业要求

题目：设计出一款简单32位cpu，至少有8条指令。

要求包括：

1. 指令的机器码的定义。
2. 运行器和移位寄存器的设计。
3. CPU控制器的设计。
4. 指令流水线的设计。
5. CPU的整体框架图

# 二．实验内容和原理

## 1.CPU处理指令的过程

冯•诺伊曼型计算机[2]的CPU将指令和数据不加区分放在存储中，指令的处理过程需要访问存储。单周期 CPU 指的是一条指令的执行在一个时钟周期内完成，然后开始下一条指令的执行，即一条指令用一个时钟周期完成。电平从低到高变化的瞬间称为时钟上升沿，两个相邻时钟上升沿之间的时间间隔称为一个时钟周期。时钟周期一般也称振荡周期（如果晶振的输出没有经过分频就直接作为 CPU 的工作时钟，则时钟周期就等于振荡周期。若振荡周期经二分频后形成时钟脉冲信号作为 CPU 的工作时钟，这样，时钟周期就是振荡周期的两倍。）

CPU 在处理指令时，一般需要经过以下几个步骤：

(1) 取指令(IF)：根据程序计数器 PC 中的指令地址，从存储器中取出一条指令，同时，PC 根据指令字长度自动递增产生下一条指令所需要的指令地址，但遇到“地址转移”指令时，则控制器把“转移地址”送入 PC，当然得到的“地址”需要做些变换才送入 PC。

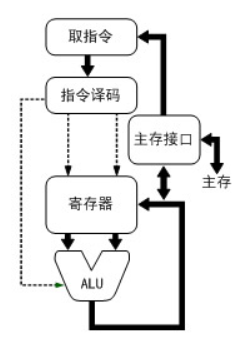
(2) 指令译码(ID)：对取指令操作中得到的指令进行分析并译码，确定这条指令需要完成的操作，从而产生相应的操作控制信号，用于驱动执行状态中的各种操作。

(3) 指令执行(EXE)：根据指令译码得到的操作控制信号，具体地执行指令动作，然后转移到结果写回状态。

(4) 存储器访问(MEM)：所有需要访问存储器的操作都将在这个步骤中执行，该步骤给出存储器的数据地址，把数据写入到存储器中数据地址所指定的存储单元或者从存储器中得到数据地址单元中的数据。

(5) 结果写回(WB)：指令执行的结果或者访问存储器中得到的数据写回相应的目的寄存器中。

单周期 CPU，是在一个时钟周期内完成这五个阶段的处理。



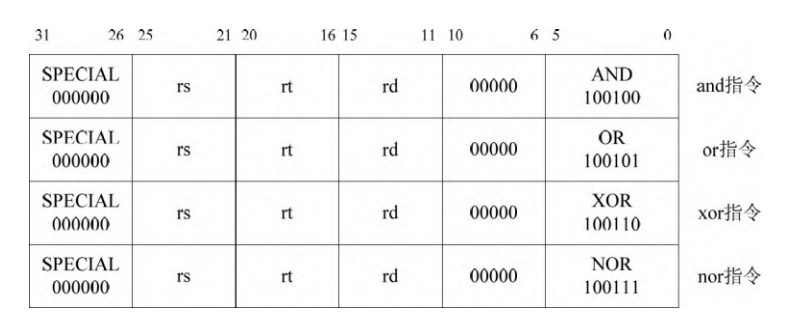
**图1 指令的典型处理过程**

## 实现的MIPS指令

### 2.1逻辑操作指令

1. and、or、xor、nor

这4条指令的格式如图2所示，从图中可以发现，这4条指令都是R类型指令，并且指令码都是6'b000000，也就是MIPS32指令集架构中定义的SPECIAL类。此外，第6～10bit都为0，需要依据指令中第0～ 5bit功能码的值进一步判断是哪一种指令。



**图2 R类型逻辑操作指令**

* and指令，逻辑“与”运算。功能码是6'b100100

**指令用法**：and rd, rs, rt

**指令作用**：rd <- rs AND rt，将地址为rs的通用寄存器的值与地址为rt的通用寄存器的值进行逻辑“与”运算，运算结果保存到地址为rd的通用寄存器中。

* or指令，逻辑“或”运算。功能码是6'b100101

**指令用法**：or rd, rs, rt

**指令作用**：rd <- rs OR rt，将地址为rs的通用寄存器的值与地址为rt的通用寄存器的值进行逻辑“或”运算，运算结果保存到地址为rd的通用寄存器中。

* xor指令，“异或”运算。 功能码是6'b100110

**指令用法**：xor rd, rs, rt

**指令作用**：rd <- rs XOR rt，将地址为rs的通用寄存器的值与地址为rt的通用寄存器的值进行逻辑“异或”运算，运算结果保存到地址为rd的通用寄存器中。

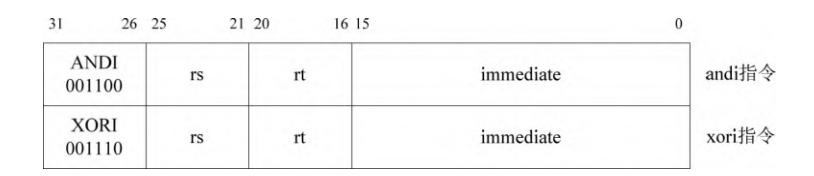
* nor指令，“或非”运算。功能码是6'b100111

**指令用法**：nor rd, rs, rt

**指令作用**：rd <- rs NOR rt，将地址为rs的通用寄存器的值，与地址为rt的通用寄存器的值进行逻辑“或非”运算，运算结果保存到地址为rd的通用寄存器中。

1. andi、xori

这2条指令的格式如图3所示，从图5-11中可以发现这2条指令都是I类型指令，可以依据指令中第26～31bit指令码的值判断是哪一种指令。



**图3 I类型逻辑操作指令**

* andi指令，逻辑“与”运算。 指令码是6'b001100

**指令用法**：andi rt, rs, immediate

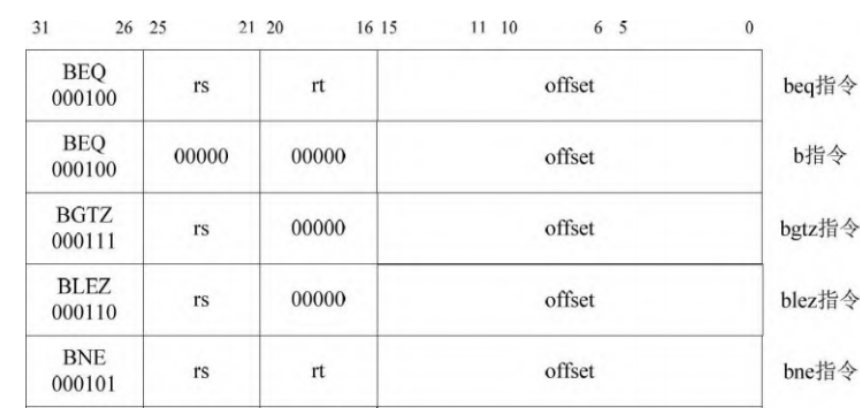
**指令作用**：rt <- rs AND zero\_extended(immediate)，将地址为rs的通用寄存器的值与指令中立即数进行零扩展后的值进行逻辑“与”运算，运算结果保存到地址为rt的通用寄存器中。

* xori指令，异或运算。 指令码是6'b001110

**指令用法**：xori rt, rs, immediate

**指令作用**：rt <- rs XOR zero\_extended(immediate)，将地址为rs的通用寄存器的值与指令中立即数进行零扩展后的值进行逻辑“异或”运算，运算结果保存到地址为rt的通用寄存器中。

### 2.2分支指令



**图4 分支操作指令**

* beq指令。指令码为6'b000100

**指令用法**：beq rs, rt, offset

**指令作用**：if rs = rt then branch，将地址为rs的通用寄存器的值与地址为rt的通用寄存器的值进行比较，如果相等，那么发生转移。

* b指令。指令码为6'b000100，且16-25bit为0

**指令用法**：b offset

**指令作用**：无条件转移，从图8-4可知，b指令可以认为是beq指令的特殊情况，当beq指令的rs、rt都等于0时，即为b指令，所以在MIPS实现的时候不需要特意实现b指令，只需要实现beq指令即可。

* bgtz指令。 指令码为6'b000111

**指令用法**：bgtz rs, offset

**指令作用**：if rs > 0 then branch，如果地址为rs的通用寄存 器的值大于零，那么发生转移。

* blez指令。指令码为6'b000110

**指令用法**：blez rs, offset

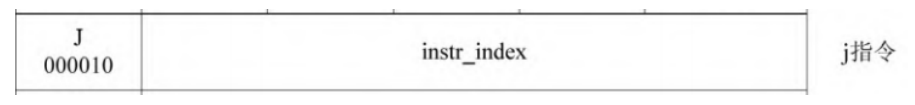
**指令作用**：if rs ≤ 0 then branch，如果地址为rs的通用寄存器的值小于等于零，那么发生转移。

* bne指令。 指令码为6'b000101

**指令用法**：bne rs, rt, offset

**指令作用**：if rs ≠ rt then branch，如果地址为rs的通用寄 存器的值不等于地址为rt的通用寄存器的值，那么发生转移。

### 2.3 跳转指令



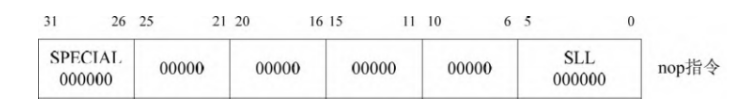
**图5 j跳转指令**

* j指令，跳转指令。指令码为6'b000010

**指令用法**：j target

**指令作用**：pc <- (pc+4)[31,28] || target || ‘00’，转移到新的指令地址，其中新指令地址的低28位是指令中的target（也就是图8-3中的instr\_index）左移两位的值，新指令地址的高4位是跳转指令后面延迟槽指令的地址高4位。

### 2.4空指令



**图6 nop空指令**

* nop指令，空指令。指令码是6'b000000

**指令用法**：nop = sll $0 $0 0

* **指令作用**：其实等价与sll指令向$0寄存器保存移位结果，实际不会有任何效果，因为无论向$0写任何数，其值始终为0，所以效果等同于什么都不做，这也正是空指令nop的效果。所以nop指令不用特意实现，完全可以当作特殊的逻辑左移指令sll。

其中，

op：为操作码；

rs：只读。为第 1 个源操作数寄存器，寄存器地址（编号）是 00000~11111，00~1F；

rt：可读可写。为第 2 个源操作数寄存器，或目的操作数寄存器，寄存器地址（同上）；

rd：只写。为目的操作数寄存器，寄存器地址（同上）；

sa：为位移量（shift amt），移位指令用于指定移多少位；

funct：为功能码，在寄存器类型指令中（R 类型）用来指定指令的功能与操作码配合使用；

immediate：为 16 位立即数，用作无符号的逻辑操作数、有符号的算术操作数、数据加载（Laod）/数据保存（Store）指令的数据地址字节偏移量和分支指令中相对程序计数器（PC）的有符号偏移量；

address：为地址。

# 三．实验过程

## （一）基本模块设计

单周期CPU每个时钟周期可被划分为五个阶段，对应地有五个最关键的模块：IF阶段对应PC，ID阶段对应InstructionMemory和RegisterFile，EXE阶段对应ALU，MEM阶段对应DataMemory，WB阶段也对应RegisterFile。除此之外，还需要生成控制信号的控制单元（ControlUnit），还要有对立即数进行扩展的ImmediateExtend。最后，还需要使用各种不同的数据选择器（Mux）构成完整的数据通路。

1. PC

PC是时序逻辑。引脚说明：

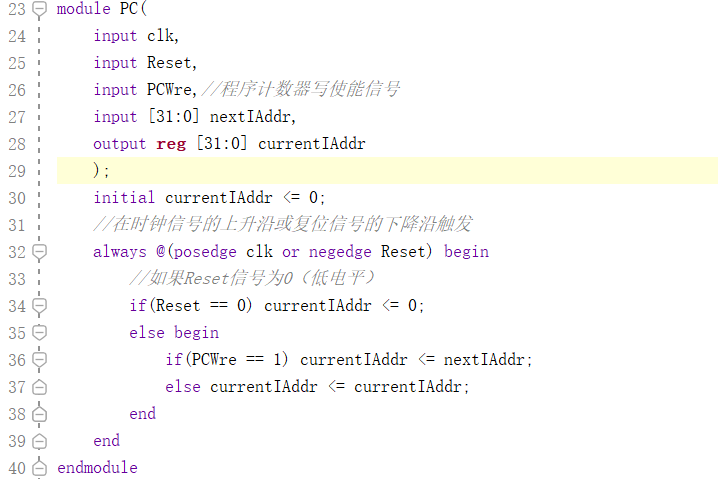
clk，输入时钟信号

Reset，复位，低电平有效，初始化PC为0

PCWre，控制信号，PC是否可以更改，0为不更改，1为更改

nextIAddr，下一条指定的地址

currentIAddr，当前正在执行的指令地址



**图7 PC模块代码**

1. InstructionMemory

InstructionMemory是组合逻辑，被设计为内含96个字节的ROM。ROM中存放

要执行的测试程序段的机器码，使用initial语句中的$readmemb伪指令从

test\_instructions.txt文件中读取出来。该外部文件的内容见附录B。该模块输入一个32位长的IAddr，输出对应地址中的32位机器指令。代码如下：

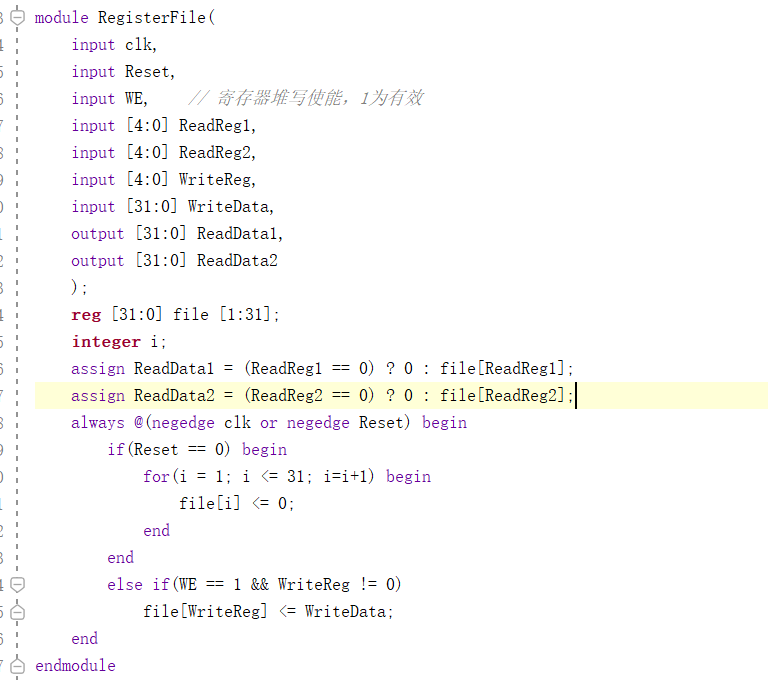


**图8 InstructionMemory模块代码**

3.RegisterFile

读寄存器堆中是组合逻辑的功能，而写寄存器堆则是时序逻辑。前者只需

用assign赋值，后者则需要clk下降沿触发写入。代码如下：



**图9 RegisterFile模块代码**

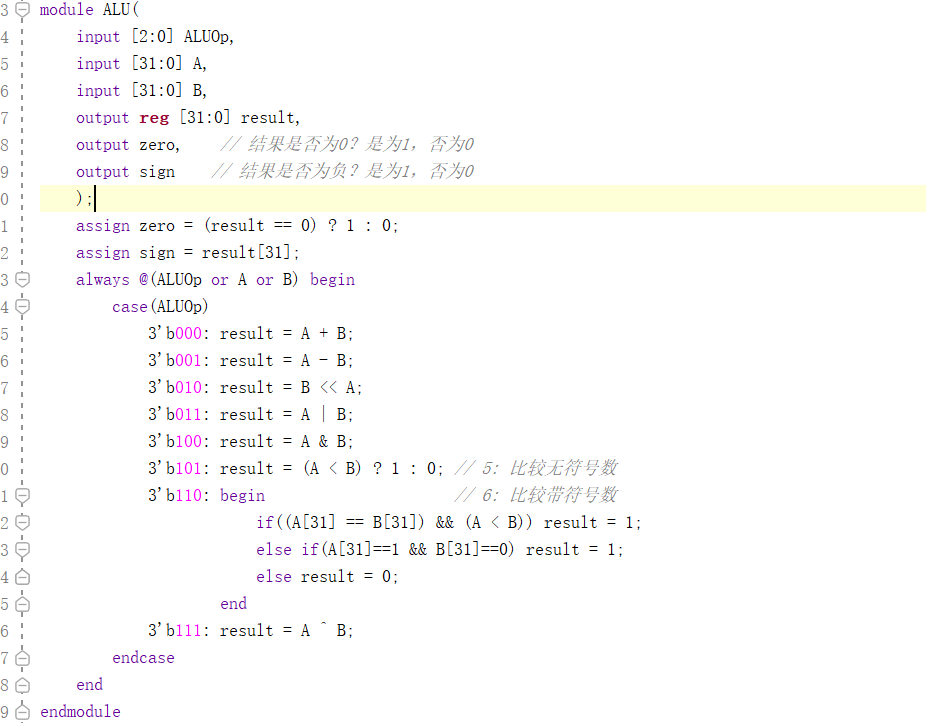
4.ALU

ALU是组合逻辑，其中：

result，ALU 运算结果

zero，运算结果标志，结果为 0，则 zero=1；否则 zero=0

sign，运算结果标志，结果最高位为 0，则 sign=0，正数；否则，sign=1，负数

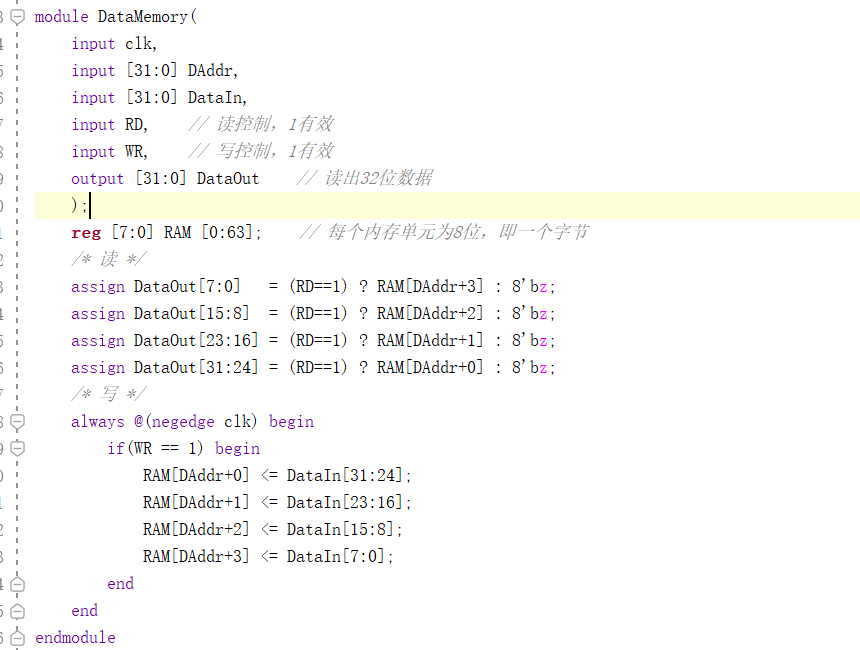


**图10 ALU模块设计**

5.DataMemory

与FileRegister类似地，读存储器为组合逻辑，写存储器为时序逻辑。前

者只需用assign赋值，后者要在clk下降沿触发写入。



**图11 DataMemory模块代码**

6.ControlUnit

ControlUnit是组合逻辑，引脚说明：

opcode，输入的6位操作码

zero，输入ALU的运算结果是否为0的指示位

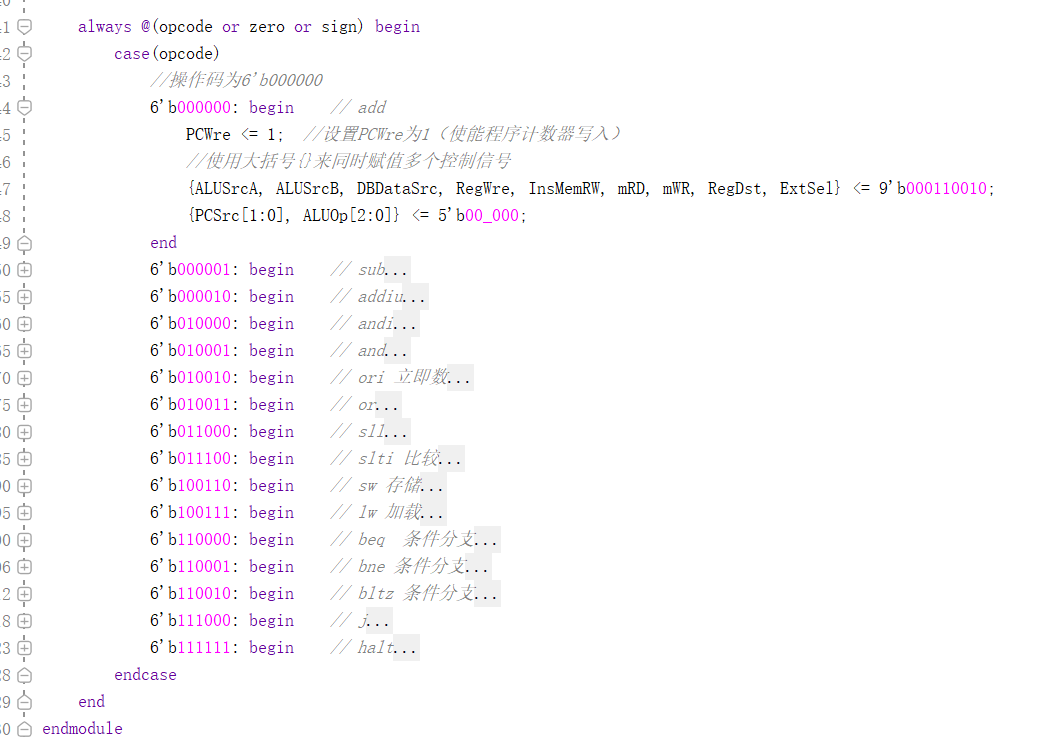
sign，输入ALU的运算结果是否为负的指示位

PCWre 、 ALUSrcA 、 ALUSrcB 、 DBDataSrc 、 RegWre 、InsMemRW、mRD、mWR、RegDst、ExtSel、PCSrc、ALUOp，共12个

控制信号，共15位。

控制单元的作用是将机器码中的操作码（opcode）转换为各个控制信号，

从而控制不同的指令在不同的数据通路中传输。



**图12 ControlUnit模块代码**

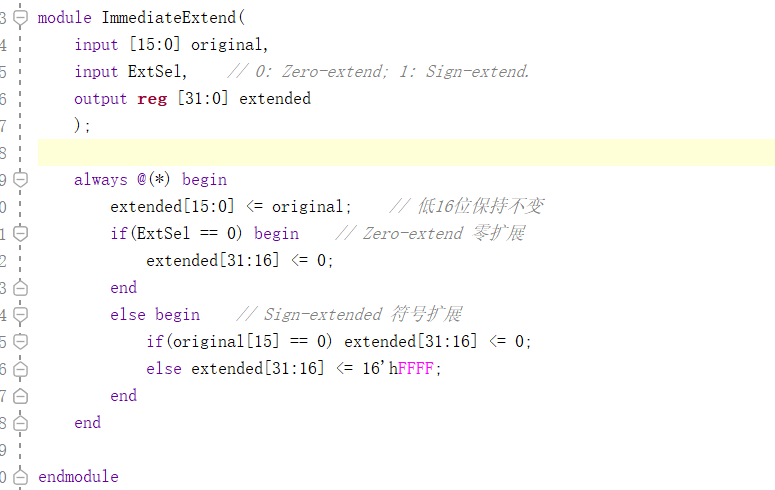
7.ImmediateExtend

该模块为组合逻辑，引脚说明如下：

original，扩展前的16位立即数

ExtSel，控制信号，决定是零扩展还是符号扩展

extended，扩展后的32位立即数



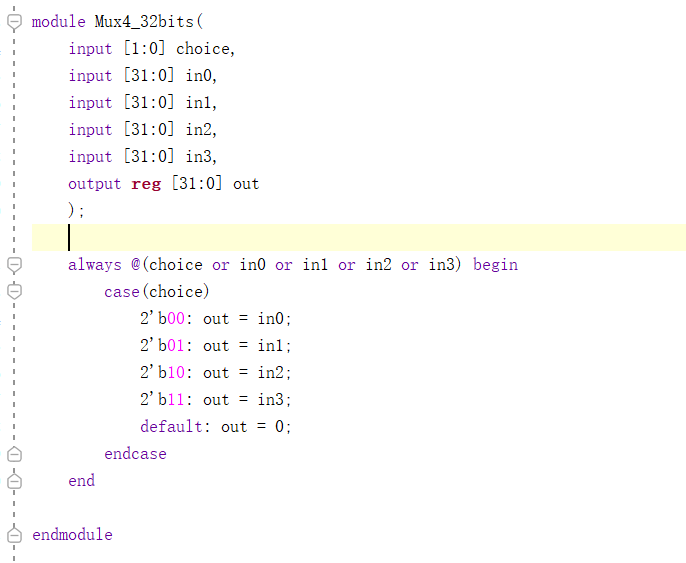
**图13 ImmediateExtend模块代码**

1. 数据选择器

根据数据通路图可知，至少需要5个数据选择器，其中有1个32位四选一选

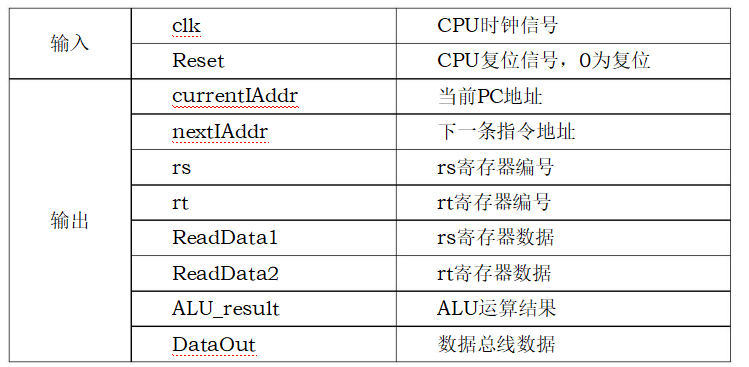
择器、1个5位二选一选择器、3个32位二选一选择器。不同的数据选择器实现

原理非常类似，因此下面只贴出32位四选一选择器的代码，其它数据选择器的代码见以Mux开头的文件。



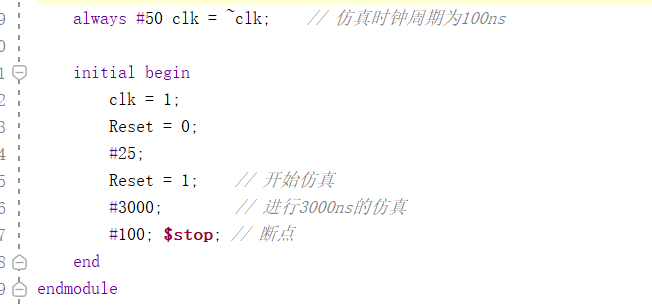
**图14 数据选择器模块代码**

## 在Vivado中仿真



**表1 top\_CPU顶层模块定义的输入输出**

仿真代码的关键部分如下：



**图15 仿真部分代码**

# 结果分析

## CPU结构

整体来看，这个电路图展示了一个简单的CPU系统，它包括输入处理、CPU核心逻辑、输出控制以及与外部设备（如LED或显示器）的接口。

其中：

Button\_Debounce：这是一个按钮去抖动模块，用于处理输入按钮信号，确保输入信号稳定，避免因为按钮接触不良导致的信号抖动。

top\_CPU：这是CPU的核心部分，包含多个寄存器和ALU（算术逻辑单元）。

For\_LED：这个模块可能用于控制LED的显示，它接收来自CPU的信号，并根据这些信号控制LED的状态。

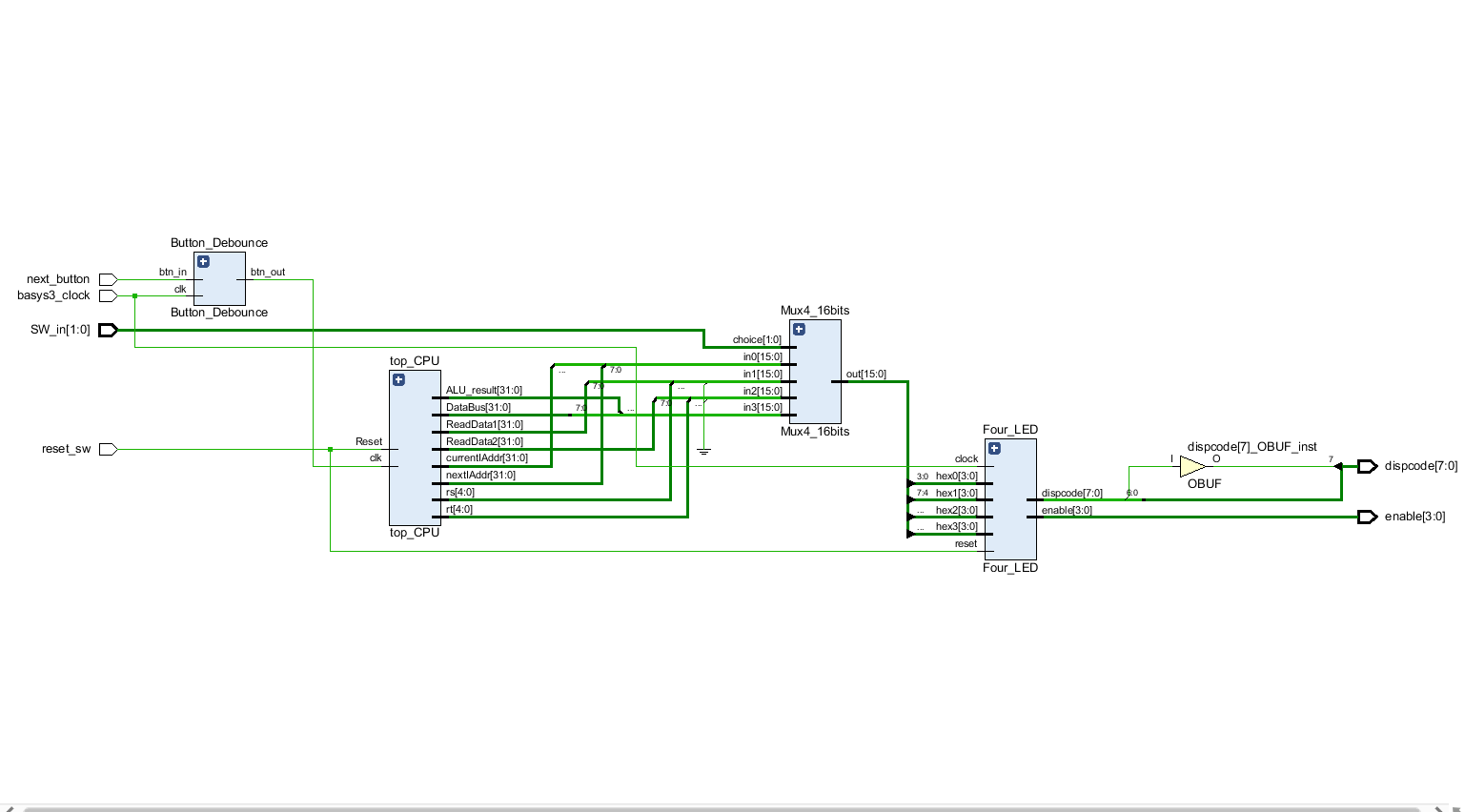
OBUF：这是一个输出缓冲器，用于将信号输出到外部设备，如LED。

dispcode[7:0] 和 enable[3:0]：这些信号可能用于控制一个显示器或LED阵列，dispcode可能表示要显示的字符代码，而enable可能用于启用或禁用特定的显示行或列。

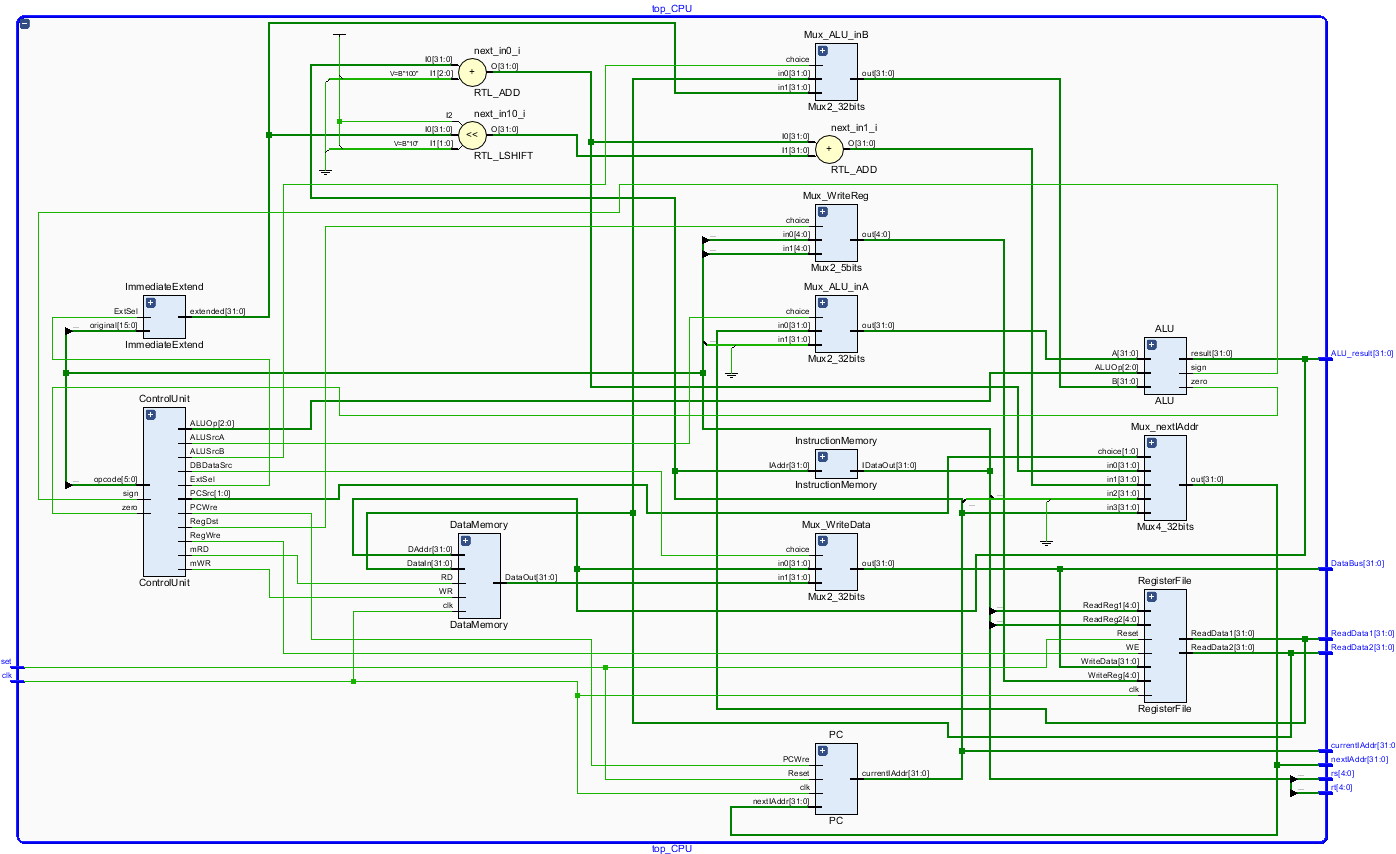
Reset：这是一个复位信号，用于初始化CPU和相关电路。

clk：这是时钟信号，用于同步电路中的所有操作。

next\_3w 和 btn\_in：这些可能是用于控制或配置CPU操作的额外输入信号。



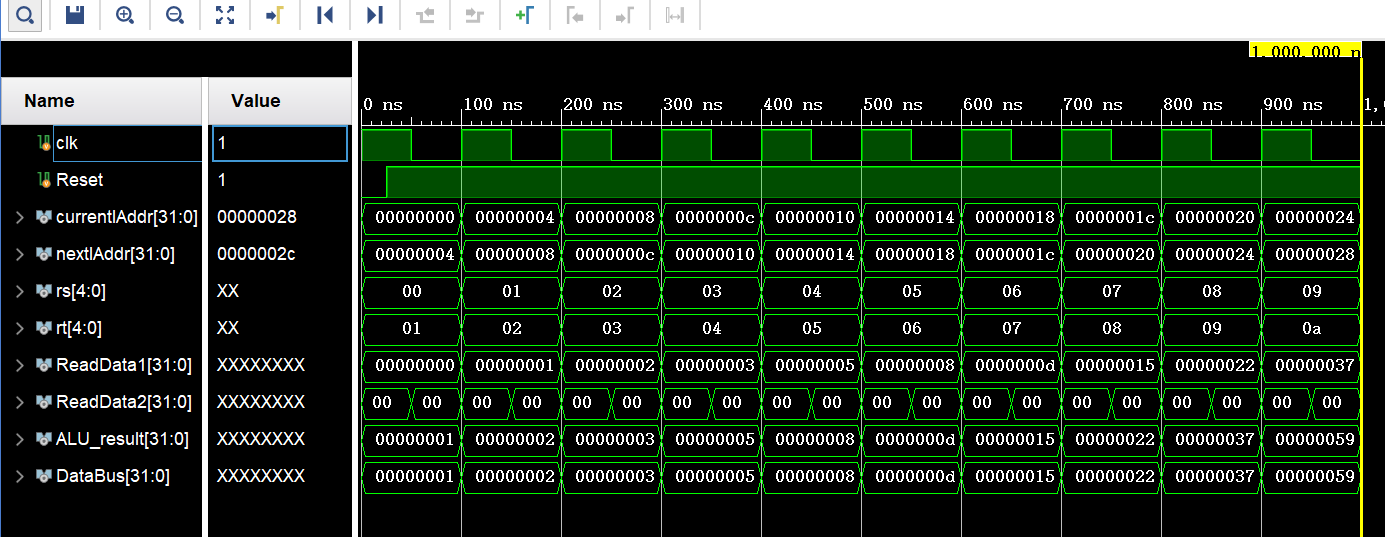
**图16 CPU整体架构**



**图17 CPU内部结构**

图片展示了CPU的数据路径和控制单元的示意图，包括ALU（算术逻辑单元），寄存器文件（Register File）多路选择器（Mux），程序计数器（PC），控制单元（Control Unit）立即数扩展（Immediate Extend），指令解码（Instruction Decode），数据存储器（Data Memory），指令存储器（Instruction Memory），ALU操作控制（ALUOp），复位（Reset），时钟（Clock）等模块。展示了CPU内部的各个组件如何通过数据路径和控制信号相互连接，以执行指令和处理数据。

## 仿真结果分析



**图18 仿真波形图**

从波形图中可以看出，随着时钟的推进，currentAddr和nextAddr在递增，ReadData1和ReadData2在变化，ALUOut在根据前两个数计算新的斐波那契数，而DataBus则将这些计算结果传输到内存中。这个过程重复进行，生成了一系列的斐波那契数。

# 总结

首先得培养全局观念。单周期CPU设计这一项实验作业是一项比较大的工程项目，因此，在开始实验之前，必须要有全局观，要在脑海中构建出该实验的大致框架：熟悉CPU内部结构、熟悉单周期CPU的数据通路图、理解PC、指令存储器、寄存器堆、ALU、数据存储器的工作原理，还要清楚地了解实验内容中要求的每一条指令的组成。

更加深入地理解了Verilog语言的“模块化”这一特性。本次设计的单周期CPU，其模块可以分为3个级别：最顶层的Basys3\_CPU，其中包含top\_CPU和Four\_LED两个模块，再往下则是各个底层模块。通过这次课程设计，我不仅提升了在数字电路和计算机体系结构领域的综合应用能力，还深刻理解了RISC-V指令集的设计理念及其在现代计算机体系结构发展中的重要性。我也学会了如何更好地管理复杂性，使设计更清晰可维护。