# 目 录

[目 录 I](#_Toc510713843)

[1 实验目的和意义 1](#_Toc510713844)

[1.1 实验目的 1](#_Toc510713845)

[1.2 实验意义 1](#_Toc510713846)

[2 实验设计 2](#_Toc510713847)

[2.1 概述 2](#_Toc510713848)

[2.2 实验环境 2](#_Toc510713849)

[2.2.1 Verilog HDL简介 2](#_Toc510713850)

[2.2.2 ModelSim简介 2](#_Toc510713851)

[2.2.3 MARS 简介 2](#_Toc510713852)

[2.2 硬件设计 3](#_Toc510713853)

[2.2.1 CPU总体结构 3](#_Toc510713854)

[2.2.2程序计数器（PC） 3](#_Toc510713855)

[2.2.3寄存器设计（RF） 3](#_Toc510713856)

[2.2.4算术逻辑运算单元（ALU） 4](#_Toc510713857)

[2.2.5扩展单元（EXT） 5](#_Toc510713858)

[2.2.6数据存储器（DM） 5](#_Toc510713859)

[2.2.7指令存储器（IM） 5](#_Toc510713860)

[2.2.8控制器（Ctrl） 6](#_Toc510713861)

[2.2.9模型机（MIPS） 7](#_Toc510713862)

[2.2.10指令地址计算单元（NPC） 7](#_Toc510713862)

[2.3 软件设计 7](#_Toc510713863)

[2.3.1添加bne指令 7](#_Toc510713864)

[2.3.2添加sll srl sllv srlv指令 8](#_Toc510713865)

[2.3.3算术逻辑运算单元（ALU） 8](#_Toc510713866)

[2.3.4扩展单元（Extender） 8](#_Toc510713867)

[2.3.5数据存储器（DMem） 8](#_Toc510713868)

[2.3.6指令存储器（IM） 8](#_Toc510713869)

[2.3.7控制器（Ctrl） 8](#_Toc510713870)

[2.3.8 模型机（MIPS） 9](#_Toc510713871)

[2.3.9 指令地址计算单元（NPC） 9](#_Toc510713871)

[2.4 设计结果分析 9](#_Toc510713872)

[2.4.1测试文件 9](#_Toc510713873)

[2.4.2测试机器码 9](#_Toc510713874)

[2.4.3 测试结果分析 9](#_Toc510713875)

[2.4.3.1 addi $t0,$zero,4 指令 12](#_Toc510713876)

[2.4.3.2 j for 指令 12](#_Toc510713877)

[2.4.3.3 lw $t5,0($t3) 指令 12](#_Toc510713877)

[2.4.3.4 beq $t7,$0,skip2 指令 12](#_Toc510713877)

[2.4.3.5 sw $2, 0($0) 指令 12](#_Toc510713877)

[2.4.3.6 bne $t7,$0,loop1 指令 12](#_Toc510713877)

[2.4.3.7 slt $t7,$t1,$t0 指令 12](#_Toc510713877)

[2.4.4 FPGA开发板测试 15](#_Toc510713875)

[2.4.4.1 FPGA开发板介绍 16](#_Toc510713877)

[2.4.4.2 FPGA开发板拍照介绍 17](#_Toc510713877)

[2.4.4.3 测试显示方案 17](#_Toc510713877)

[2.4.4.4 连接开关的代码 17](#_Toc510713877)

[2.4.4.5 灯泡和7段显示管的代码 17](#_Toc510713877)

[结论 20](#_Toc510713878)

[参考文献 20](#_Toc510713879)

[教师评语评分 21](#_Toc510713880)

# 1 实验目的和意义

## 1.1 实验目的

1. 掌握单周期CPU数据通路图的构成、原理及其设计方法；
2. 掌握单周期CPU的实现方法，代码实现方法；
3. 认识和掌握指令与CPU的关系；
4. 掌握测试单周期CPU的方法。

## 1.2 实验意义

通过自己动手在modelsim中进行指令集的设计，以及在Mars中进行排序算法的运行，通过Vavido下载到开发板上等过程，把所学的知识运用于实践，不仅加强了自己的动手操作能力，更是对于发现问题，提出问题，解决问题的思维强化。更关键的是本实验要求通过设计单周期CPU的过程，掌握Verilog HDL硬件描述语言并灵活使用，掌握用ModelSim仿真的过程，掌握Mars汇编器的调试过程，以对上学期的计算机组成原理课程有更深入的体会，提高计算机学科科研动手能力。

# 2 实验设计

## 2.1 概述

MIPS 单周期CPU设计与实现实验通过程序计数器（PC）模块、寄存器设计（RF）模块、算术逻辑运算单元（ALU）模块、扩展单元（EXT）模块、数据存储器单元（DM）模块、指令存储器（IM）模块、控制器（Ctrl）模块、指令地址计算单元（NPC）模块的设计，并在模型机（MIPS）中把各模块综合，得到完整的单周期CPU。并且用所设计的lw、sw、addu、ori、beq、lui、sll、srl、bne、j、slt、slti指令集，并且用这些指令，在Mars编写一个排序程序，实现有符号十六进制数的从小到大排序。以及通过Vavido下载到FPGA开发板上显示。本次实验设计主要遵循mips单周期的指令集、Modelsim、Mars和Vavido的使用规范。

## 2.2 实验环境

### 2.2.1 Verilog HDL简介

Verilog HDL是一种硬件描述语言（HDL:Hardware Description Language），以文本形式来描述数字系统硬件的结构和行为的语言，用它可以表示逻辑电路图、逻辑表达式，还可以表示数字逻辑系统所完成的逻辑功能。 Verilog可以将数字系统在行为级和结构级进行描述。行为级描述方法描述了数字系统的功能特性。结构级描述方法描述了数字系统的详细组织，并且结构级描述常采用层次描述。结构及描述可以在基本元件的层次结构方面描述硬件系统，比如在门级和开关级。因此，我们可以用Verilog语言来描述真值表的具体内容和数据通路。

### 2.2.2 ModelSim简介

mentor公司的ModelSim是业界最优秀的HDL语言仿真软件，它能提供友好的仿真环境，是业界唯一的单内核支持VHDL和Verilog混合仿真的仿真器。它采用直接优化的编译技术、Tcl/Tk技术、和单一内核仿真技术，编译仿真速度快，编译的代码与平台无关，便于保护IP核，个性化的图形界面和用户接口，为用户加快调错提供强有力的手段，是FPGA/ASICsheji 的首选仿真软件。

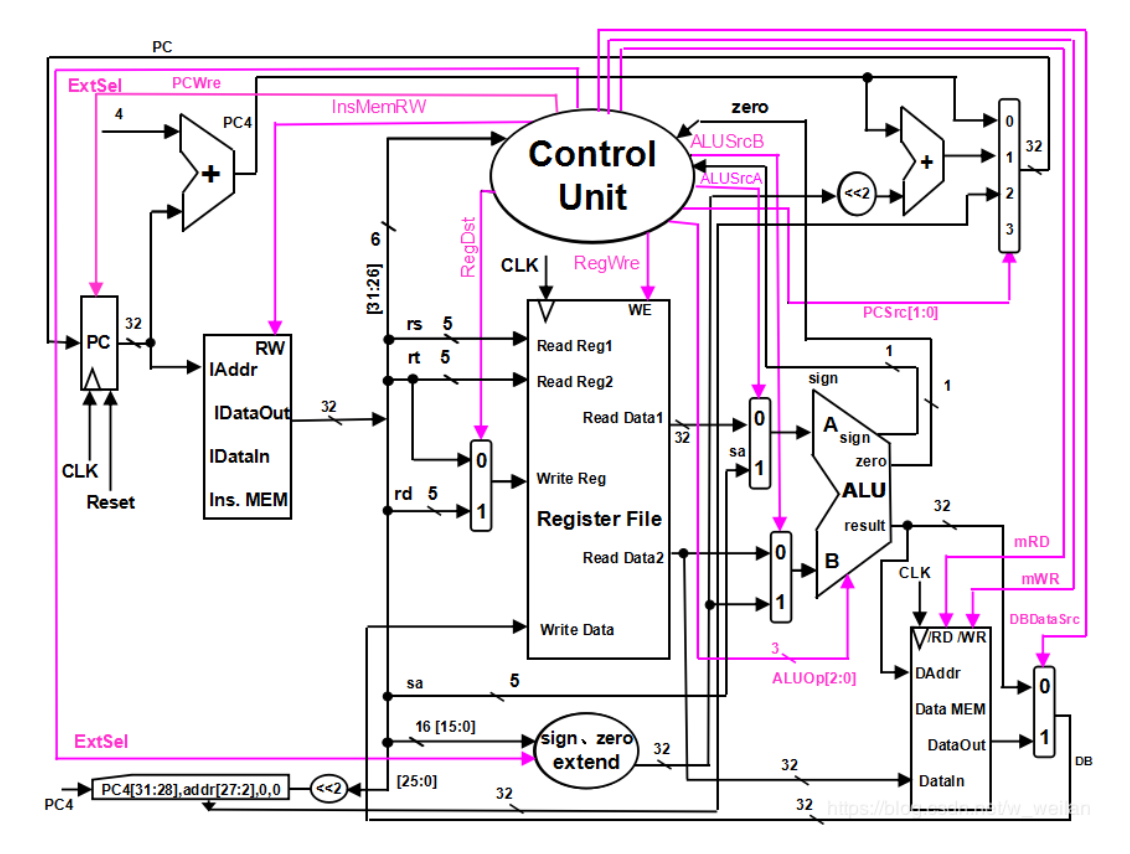
### 2.2.3 MARS 简介

MARS（MIPS汇编程序和运行时模拟器）是一个轻量级的交互式开发环境（IDE），用于使用MIPS汇编语言进行编程，旨在与Patterson和Hennessy的计算机组织和设计一起用于教育级别。

## 2.2 硬件设计

### 2.2.1 CPU总体结构

CPU总体结构如下图所示，其中包括程序计数器（PC)、指令存储器（IM）、寄存器组（RF）、运算器（ALU）、数据扩展单元（EXT）、数据存储器（DM）和控制器（Ctrl）。



### 2.2.2程序计数器（PC）

**（1）功能描述**

单周期的每个周期都会写入PC，通常情况下PC会在下一周期被更新为NPC的值。在接收到rst信号时PC会被复位为32’h00000000。

1. **模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| NPC[31:0] | I | 由NPC模块计算出来的下一条PC |
| PCWr, | I | PC更改信号 |
| clk | I | 系统时钟 |
| rst | I | 复位信号 |
| PC[31:0] | O | 取出的指令地址（字地址） |

### 2.2.3寄存器设计（RF）

**（1）功能描述**

RF为寄存器堆，clk信号控制时钟周期，RFWr信号控制寄存器是否可写。寄存器A1,A2不为0时可读出它们的值到RD1,RD2；写使能信号为1时可将WD的数据写入A3寄存器。

**（2）模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A1 | I | 需要读取的第一个寄存器的地址 |
| A2 | I | 需要读取的第二个寄存器的地址 |
| A3 | I | 需要写回的寄存器的地址 |
| WD | I | 需要写回寄存器的数据 |
| clk | I | 系统时钟 |
| RFWr | I | 寄存器写使能 |
| RD1 | O | 从寄存器读出的第一个数据 |
| RD2 | O | 从寄存器读出的第二个数据 |

### 2.2.4算术逻辑运算单元（ALU）

**（1）功能描述**

ALU主要功能是完成对输入数据的进行加法、减法、与、或、左移、右移、乘法、除法运算以及判断两个操作数是否相等。

**（2）模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| DataIn1 [31:0] | I | 操作数A |
| DataIn2 [31:0] | I | 操作数B |
| ALUOp[1:0] | I | 需要进行的运算  00：加法  01：减法  10：或运算 |
| Zero | O | 两操作数是否相等 |
| AluResult [31:0] | O | 运算结果 |

### 2.2.5扩展单元（EXT）

**（1）功能描述**

EXT主要功能是将16位的数据扩展为32位数据。

**（2）模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Imm16 [15:0] | I | 需要进行扩展的数据 |
| EXTOp[1:0] | I | 扩展方式的控制信号  00：0扩展  01：符号扩展  10：将立即数扩展到高位 |
| Imm32 [31:0] | O | 扩展结果 |

### 2.2.6数据存储器（DM）

**（1）功能描述**

用于对数据内存的读操作与写操作，定义了1024个32位的数据内存，在系统时钟的控制下，当写使能有效时，可以对指定字地址的指定字节进行写操作，也可以读取指定字地址的数据。

**（2）模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr | I | 需要进行写操作或者读操作的内存地址 |
| din | I | 需要写入内存的数据 |
| be | I | 选择需要写入的具体的某个字节 |
| DMWr | I | 数据内存的写使能信号，有效时才能对数据内存进行写操作 |
| clk | I | 系统时钟 |
| dout | O | 从数据内存中读取出来的数据 |

### 2.2.7指令存储器（IM）

**（1）功能描述**

MIPS指令代码存储的位置，为其提供指令地址后它就将指令从内存中输出。

**（2 ）模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr | I | 需要读取的指令地址 |
| dout | O | 读出的指令 |

### 2.2.8控制器（Ctrl）

**（1）功能描述**

根据输入的Op以及Funct字段，进行译码，根据不同类型的指令，产生指令执行所需要的控制信号。

**（2 ）模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 系统时钟 |
| rst | I | 复位信号 |
| Op | I | 指令的26-31位，区别指令的类型 |
| Funct | I | 指令的0-5位，只对R指令有意义，可以区别不同的R指令索要执行的操作 |
| RFWr | O | 寄存器写使能，有效时才能对寄存器进行写操作 |
| DMWr | O | 数据内存写使能，有效时才能对数据内存进行写操作 |
| EXTOp | O | 扩展信号，决定对数据进行哪一种扩展 |
| ALUOp | O | ALU信号，决定对数据进行哪一种运算 |
| GPRSel | O | 寄存器写地址，决定在WB阶段写回数据的地址是rd还是rt还是31号寄存器 |
| WDSel | O | 寄存器写数据，决定在WB阶段写回寄存器的数据是来自ALU运算的结果，还是从内存中读取出来的数据，还是下一条指令的地址 |
| BSel | O | 决定ALU运算的第二源操作数是来自从寄存器读取出来的数据，还是经过扩展的32位立即数 |
| NPCOp | O | 决定下一步要取的指令地址是当前指令地址加4，还是分支跳转地址，还是跳转指令地址，还是从寄存器读取出来的间接跳转的地址 |

### 2.2.9单周期CPU（SCCPU）

**（1）功能描述**

设计的顶层文件，把单周期CPU的各个部件连接起来。

**（2 ）模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Clk | I | 系统时钟 |
| Rst | I | 复位信号 |

### 2.2.10指令地址计算单元（NPC）

**（1）功能描述**

根据控制模块译码得到的指令类型，计算下一条指令的地址是在原有的指令地址上加一，还是条件分支，还是跳转指令，还是跳转至通用寄存器保存的地址。计算出来的新地址将传递给PC，在IF阶段取下一条指令

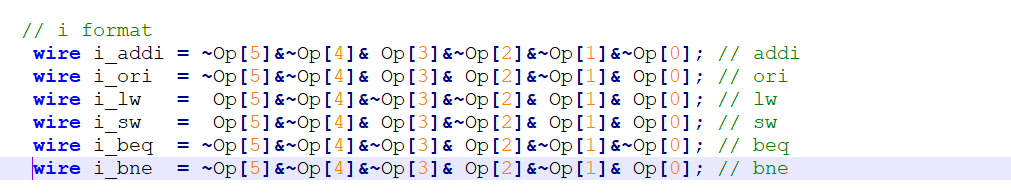
**（2 ）模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| PC | I | 目前的指令地址 |
| NPCOp | I | 根据指令类型得到的新地址计算方式  NPC\_PLUS4 2'b00  NPC\_BRANCH 2'b01  NPC\_JUMP 2'b10  NPC\_JR 2'b11 |
| IMM | I | 来自指令0-15位的立即数 |
| RD | I | 31号通用寄存器保存的数据 |
| NPC | O | 计算出来的新指令地址 |

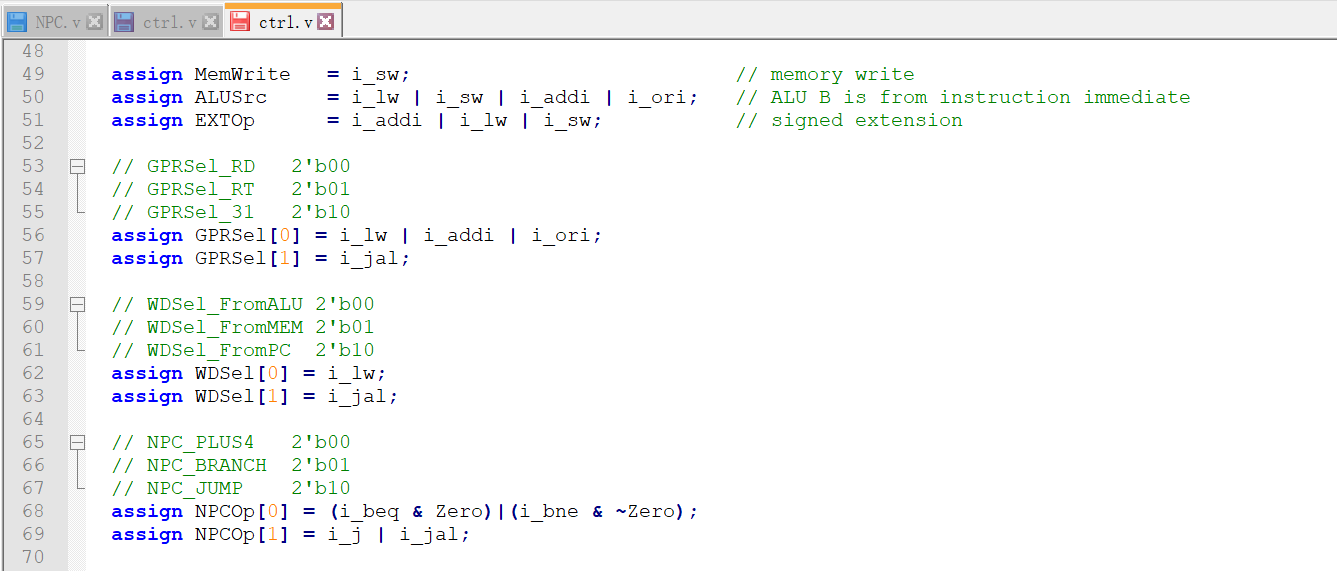
## 2.3 软件设计

### 2.3.1添加bne指令

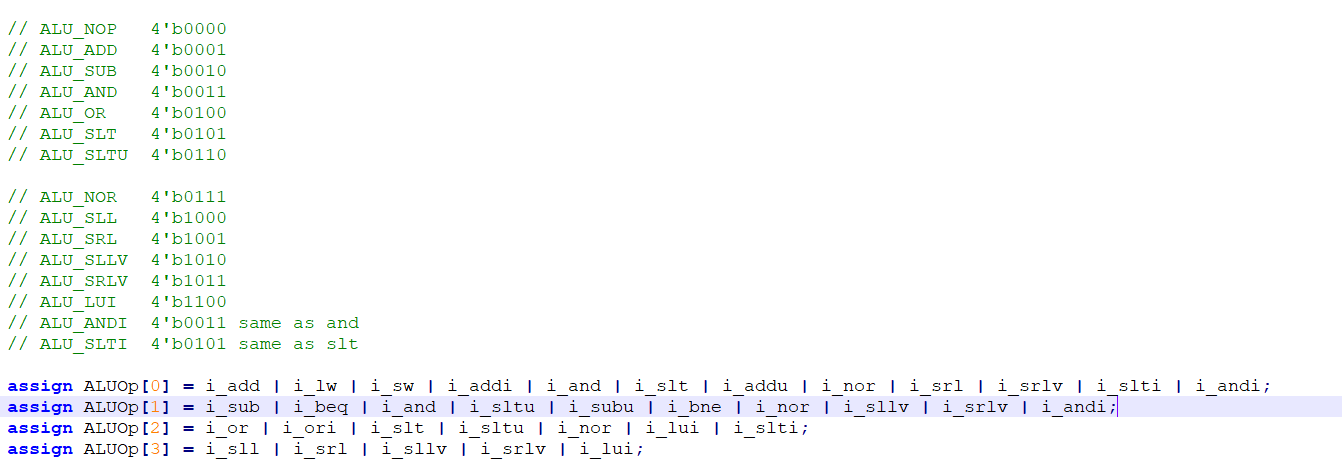
1、增添指令译码



2、因为bne需要跳转，所以要修改NPCOp

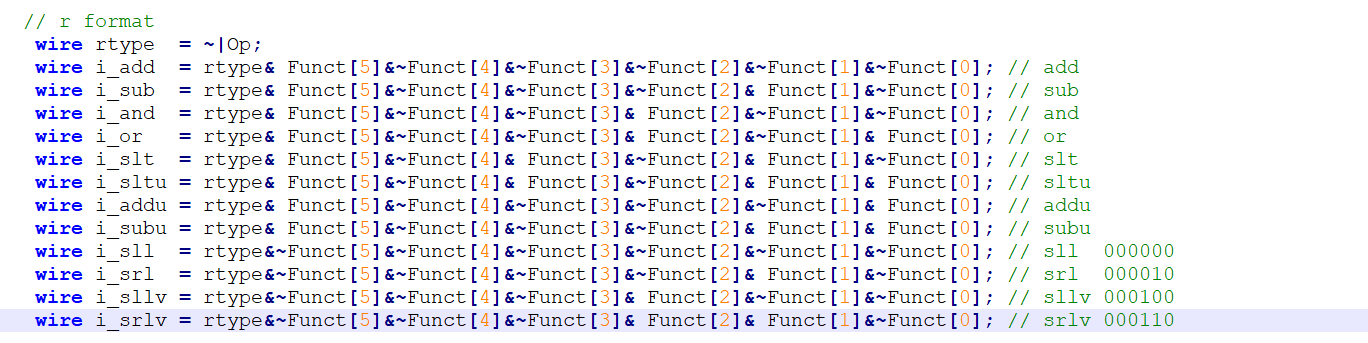


3、用到alu减法运算，故添加ALUOp

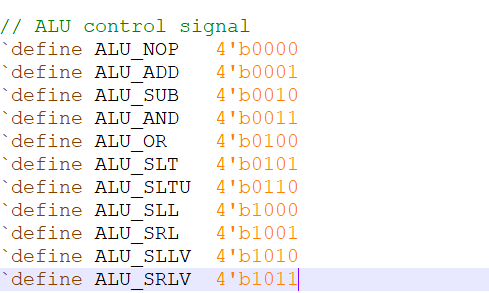


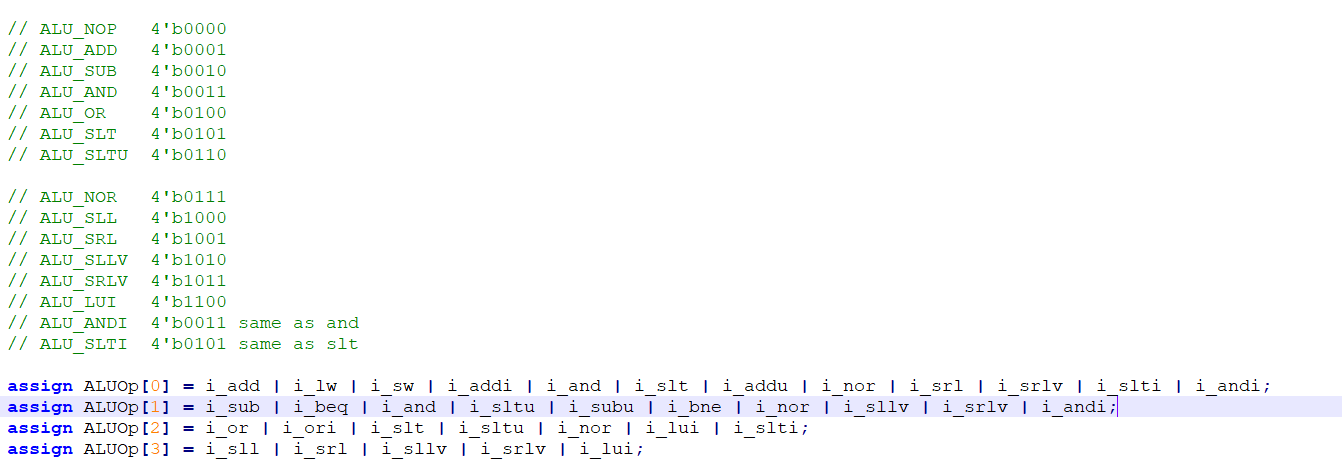
### 2.3.2添加sll & srl & sllv & srlv指令

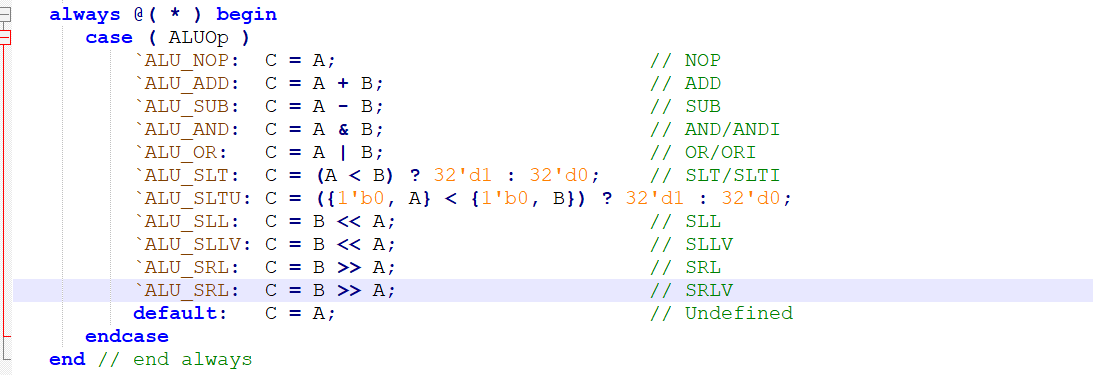
1、首先添加指令译码



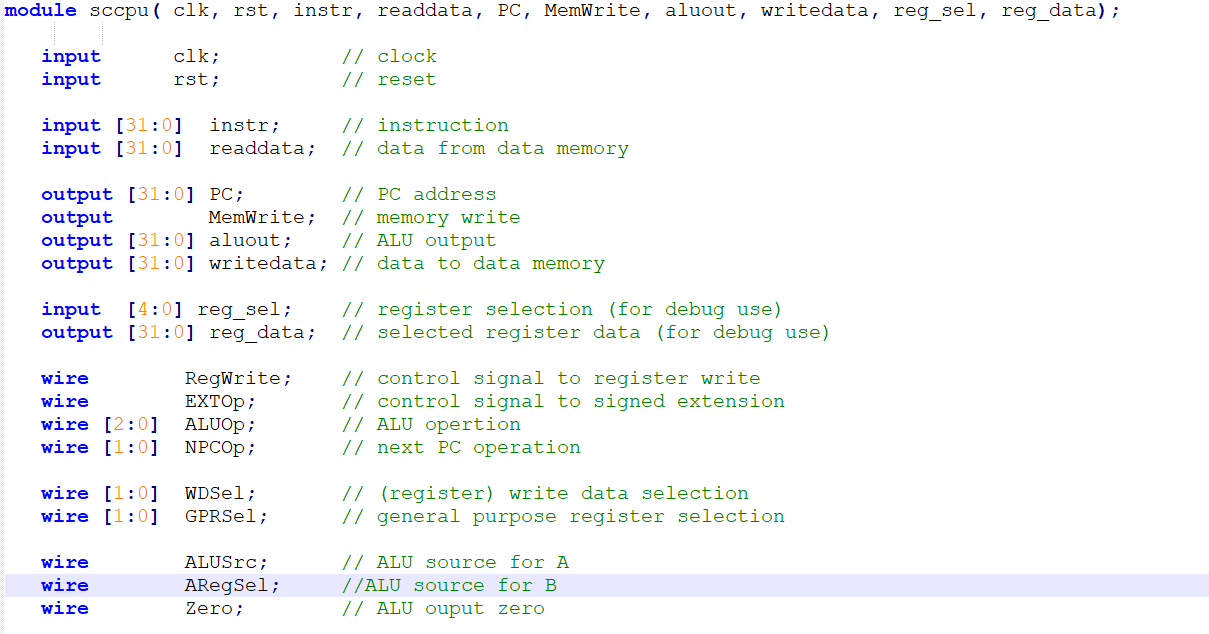
2、拓展ALU



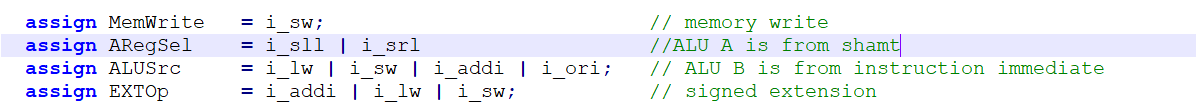




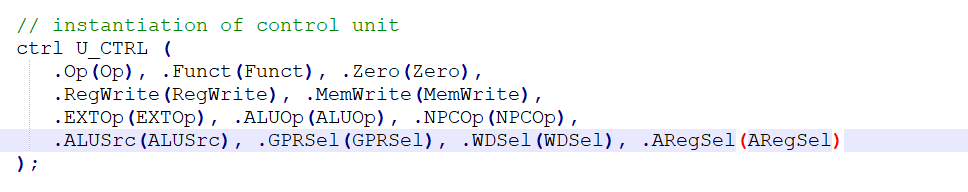
3、添加ARegSel信号（需要添加多路选择器，ALUSrcA来自于shamt，ALUSrcB来源于imm）



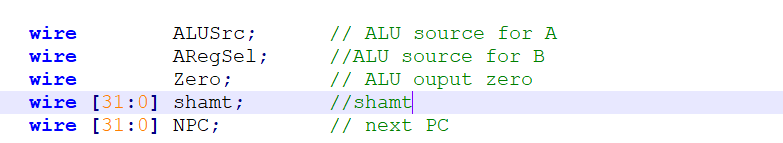
4、为ARegSel赋值



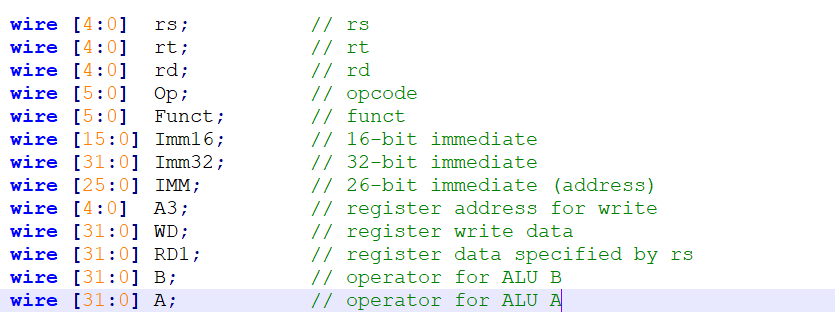
5、绑定信号



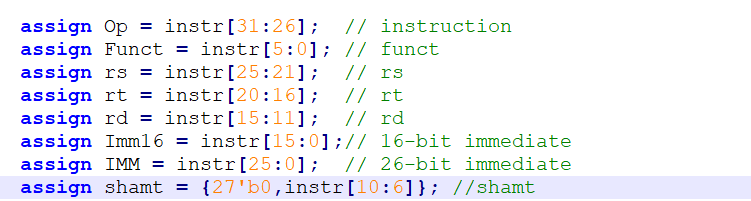
6、补充shamt信号



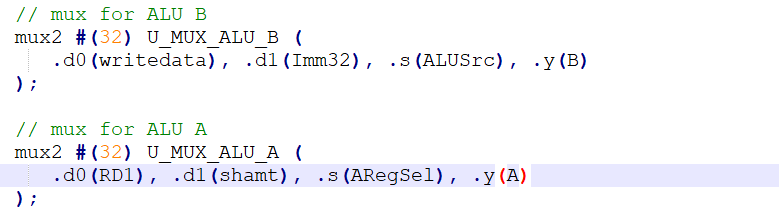
7、补充ALU A信号



8、定义shamt，高27位为0，低5位为instr的6到10位

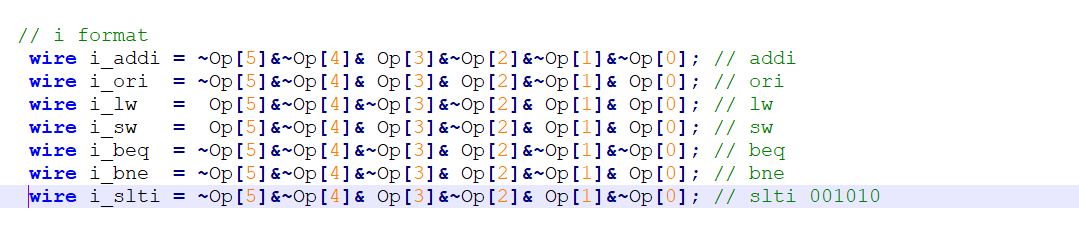


9、补充多路选择器

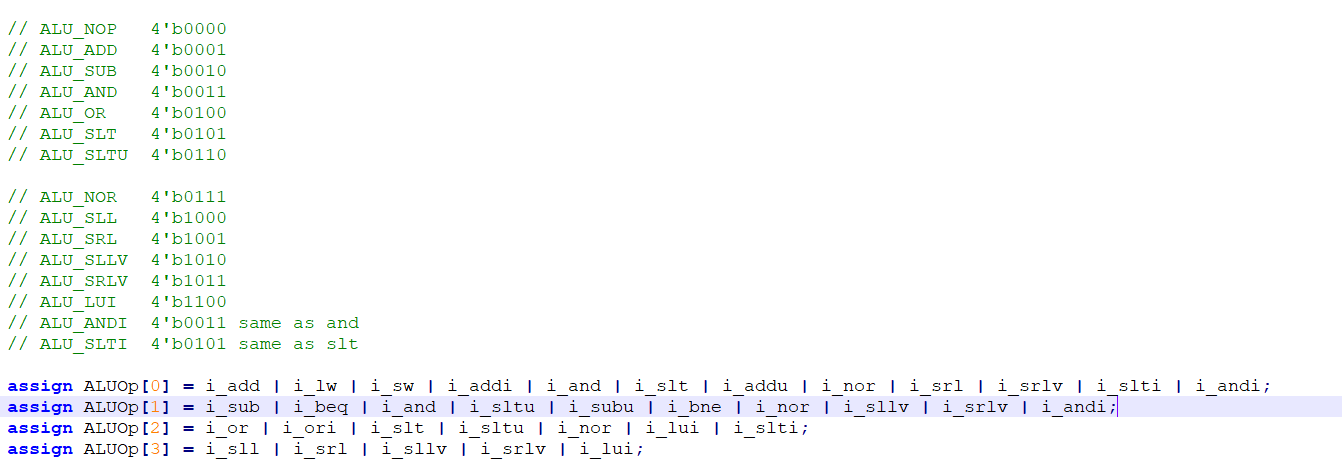


### 添加slti（小于立即数则置位）I型指令

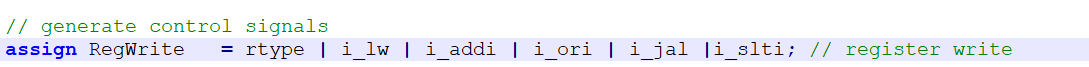
1、添加指令译码



2、补充ALUOp（同slt）

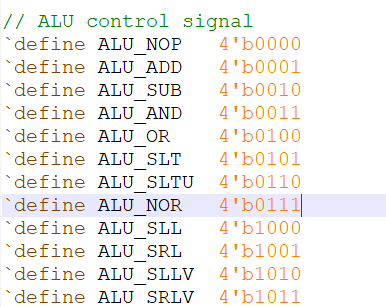


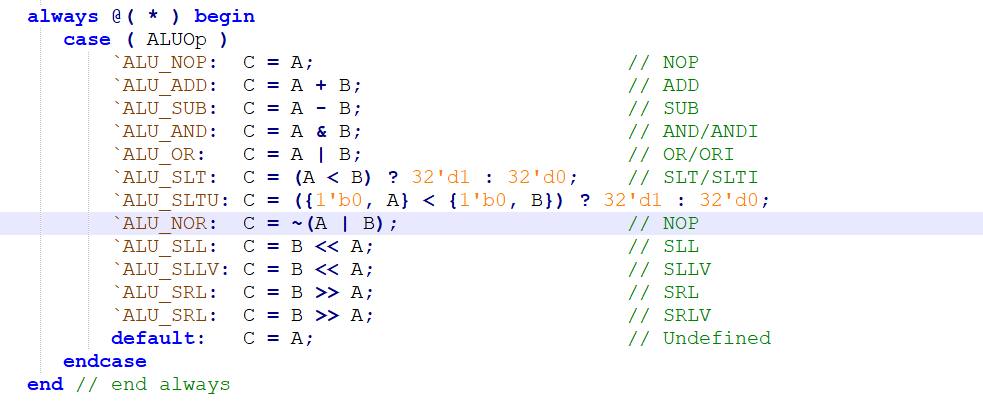
3、补充寄存器写操作

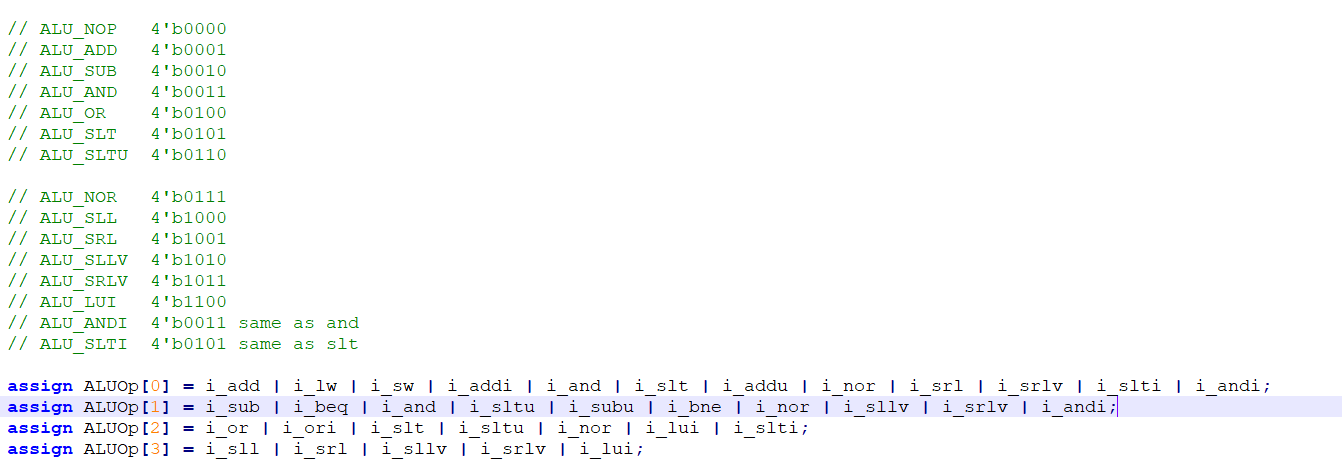


### 添加nor（异或）R型指令

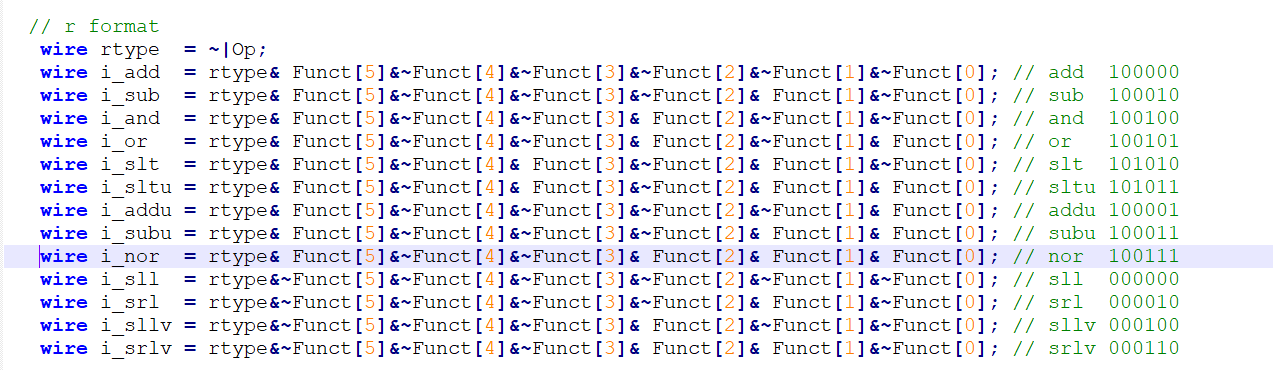
1、扩充ALU





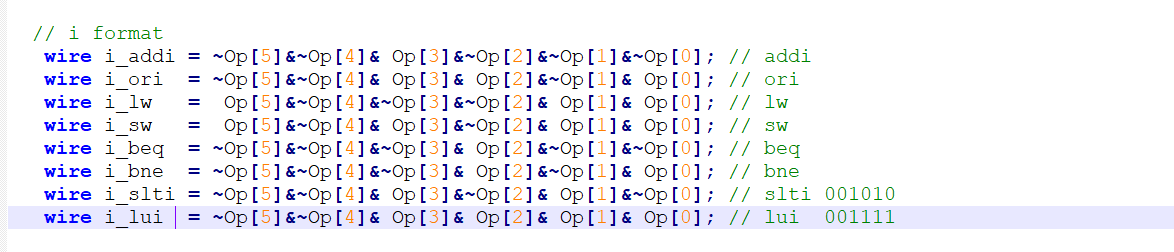


2、添加指令译码

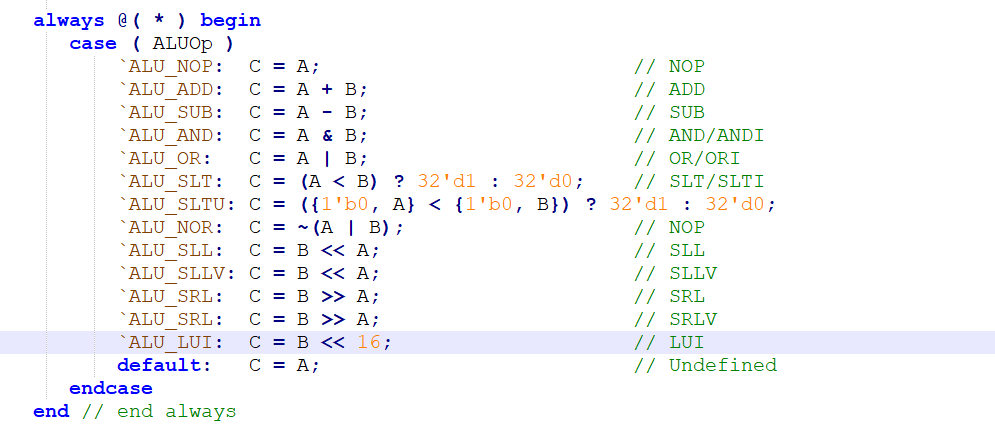


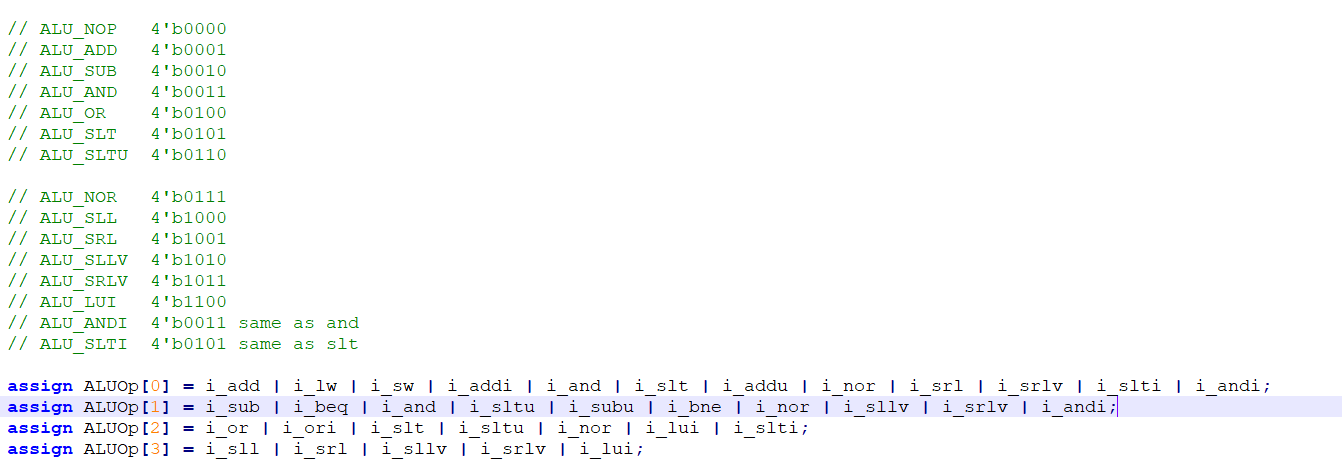
### 添加lui指令

1. 添加指令译码

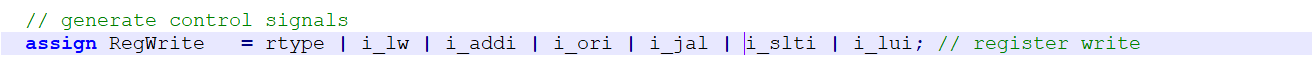


1. ALU扩展

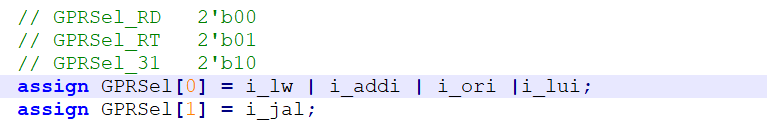




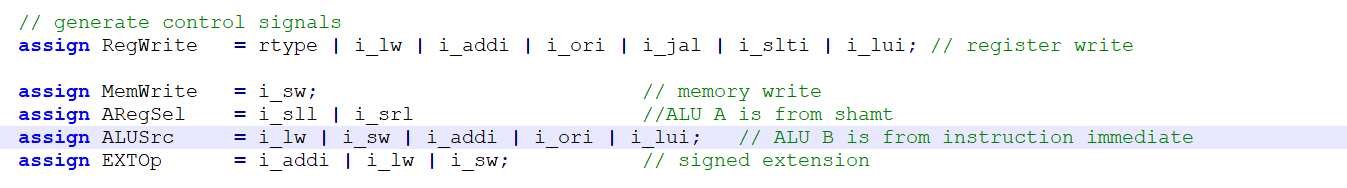
1. 修改写寄存器



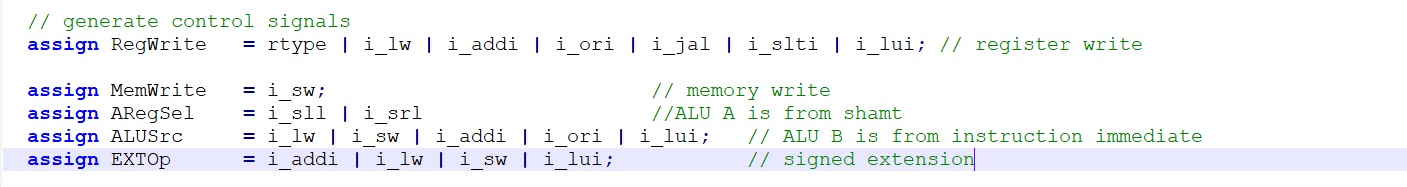
4、写回rt寄存器



1. ALU运算，来自一个立即数

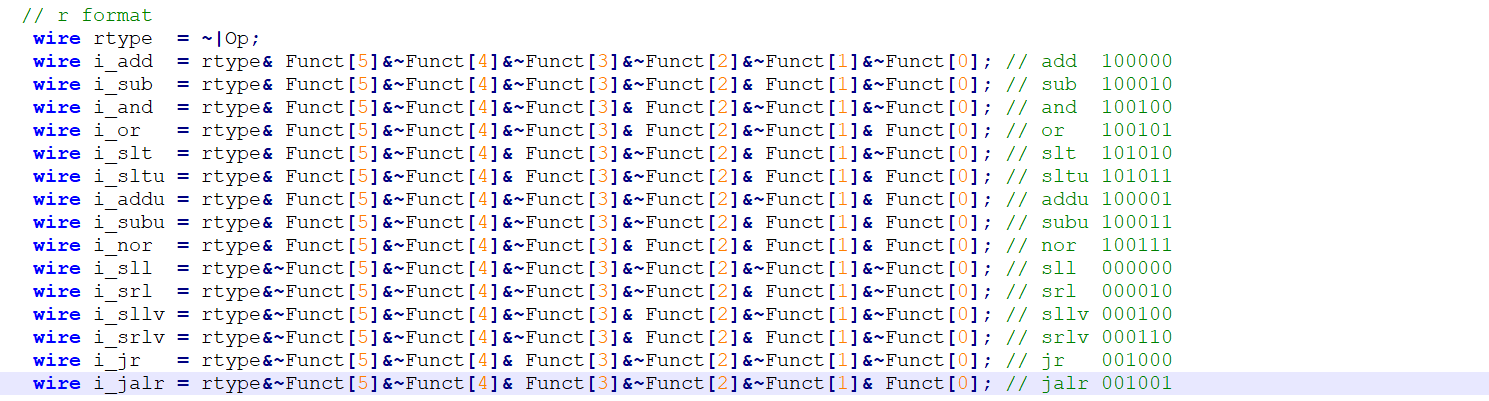


1. 符号位拓展

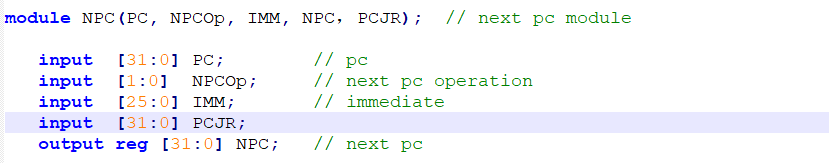


### 添加jr指令

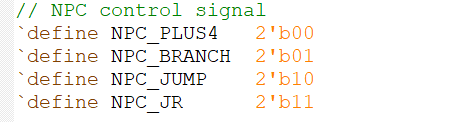
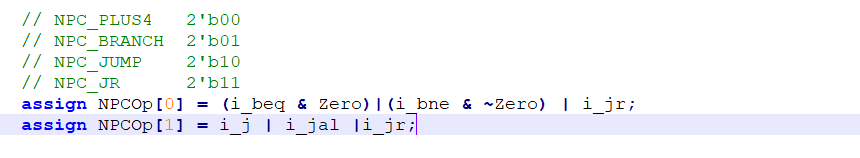
1. 添加指令译码

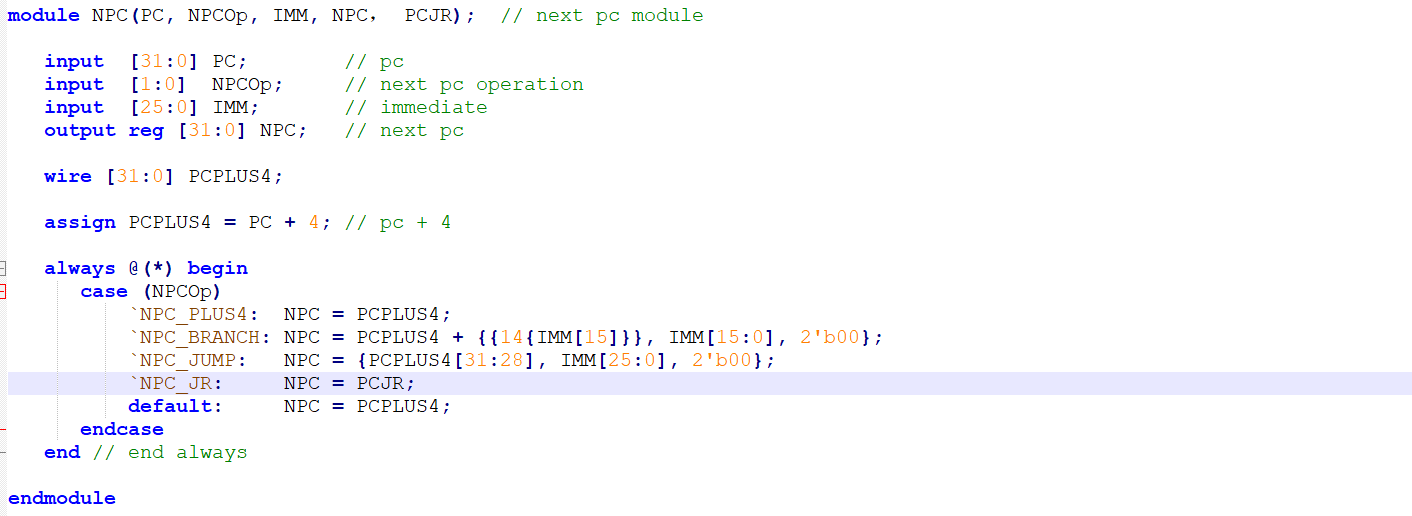


1. 添加PCJR信号并绑定引脚

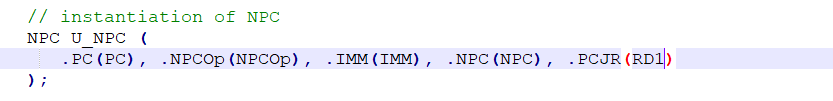


3、扩充NPC

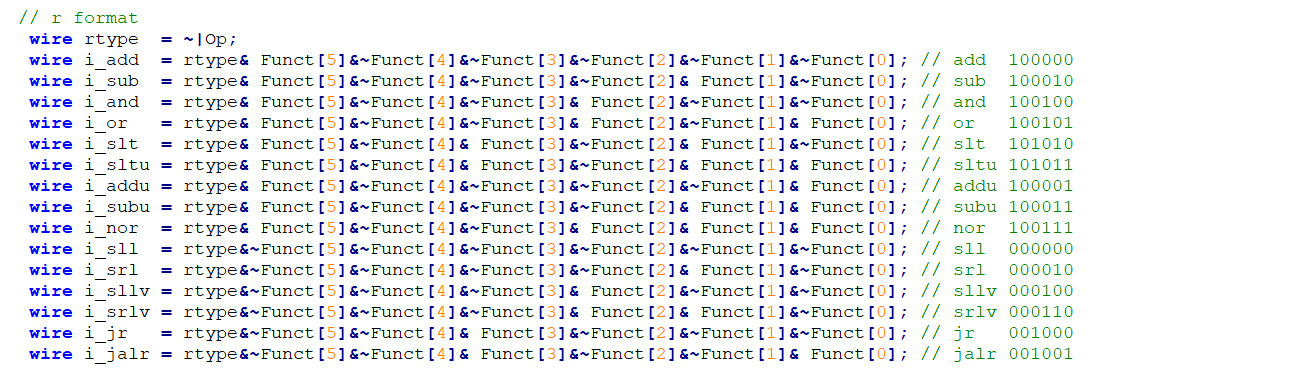


1. 和寄存器的输出绑定

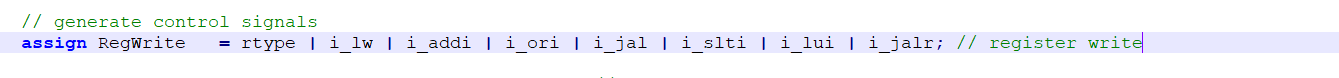


### 添加jalr指令

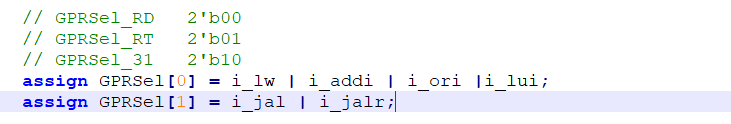
1. 指令译码



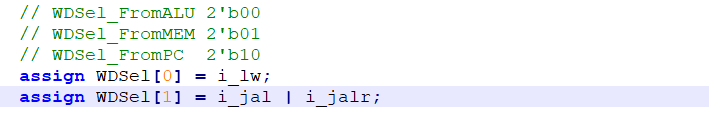
1. 添加写寄存器



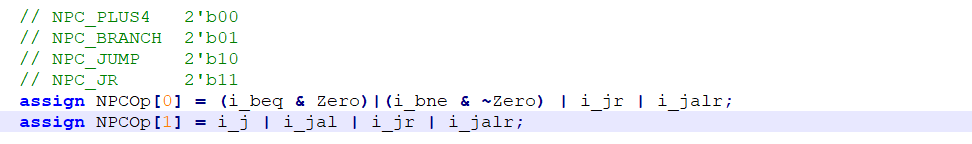
1. 选择寄存器



4、选择需要写的数据（来自PC）

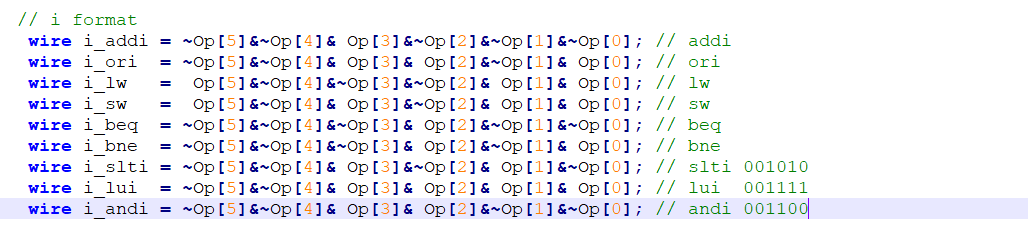


5、选择NPC

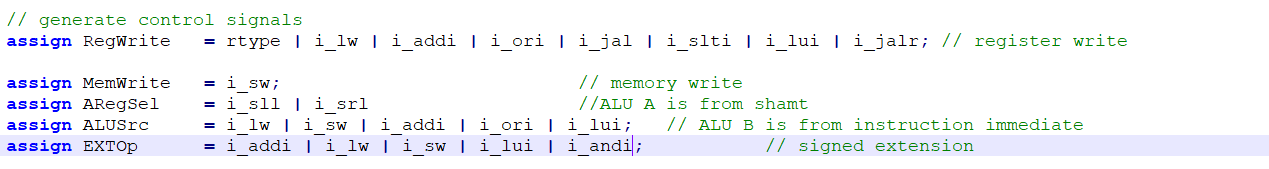


### 添加andi指令

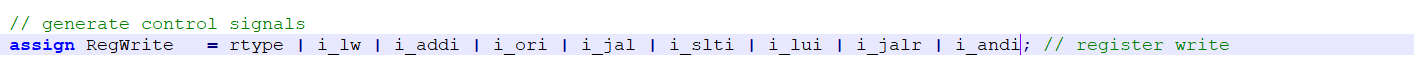
1. 指令译码



1. 需要进行符号拓展



1. 需要写寄存器



1. 扩充ALU

