**编译器设计**

**实验指导书**

**任课老师：潘志铭**



作为一种开源指令集，RISC-V的特点是模块化设计、扩展性强，当标准的RISC-V扩展指令无法满足客户需求时，就需要在RISC-V规范允许范围内增加自定义指令集。当我们在硬件上实现了这些自定义指令后，为了使用这些自定义指令集，我们还需要修改工具链，本文将举例说明如何为GNU工具链增加RISC-V自定义指令的支持。在修改GNU工具链之前，我们需要预先下载riscv-gnu-toolchain和riscv-tools，具体步骤为：

$ git clone <https://github.com/riscv/riscv-gnu-toolchain.git> --recursive

$ git clone <https://github.com/riscv/riscv-tools.git> --recursive

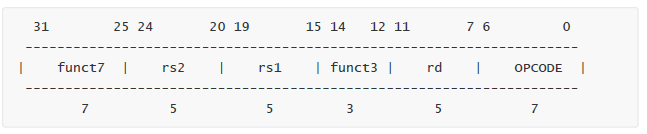
1. **描述需要增加的自定义指令**

假定我们需要增加一条乘加指令**mac**，具体描述如下：

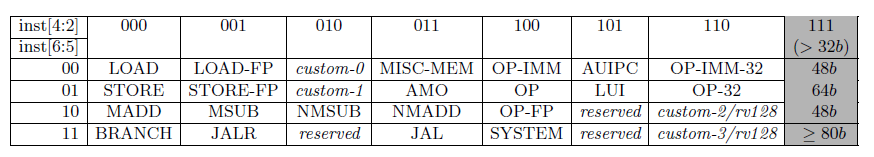
|  |  |
| --- | --- |
| 汇编指令 | mac rd, rs1, rs2 |
| 软件模型 | rd = rd + rs1 \* rs2 |
| 例子 | mac a2, a0, a1 |

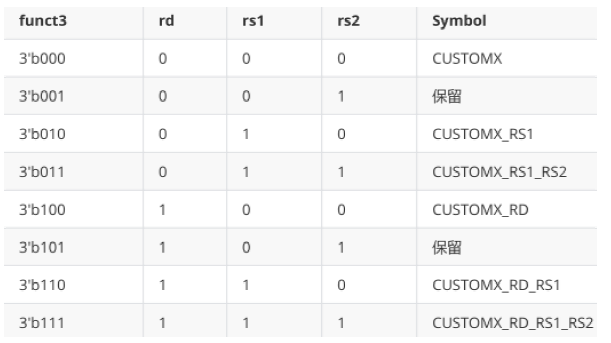
1. **确定指令码**

依据RISC-V SPEC的描述，**mac**指令属于R-Type指令，所以指令码大致格式为：

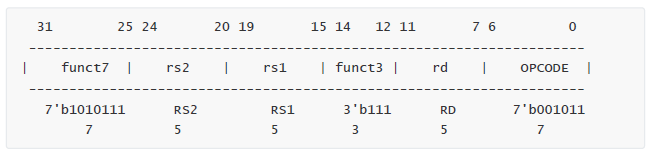


RISC-V SPEC规定了四种自定义指令的OPCODE，分别是7‘b000\_1011，7’b010\_1011，7’b101\_1011，7’b111\_1011，这里我们选用CUSTOM0编码7’b000\_1011，为了进一步规范自定义指令码格式，我们可以约定funct3的编码格式：3’b010表示只使用了RS1寄存器，3’b011表示只使用了RS1和RS2寄存器，3’b100表示只使用了RD寄存器，3’b110表示只使用RD和RS1寄存器，而3’b111则表示RD、RS1和RS2寄存器都用到了，所以这里的funct3应该是3’b111.由于funct7没有特别的约束，我们这里选定为7’b101\_0111.





最终选定的编码如下：



1. **为binutils增加自定义指令**
   1. 编写**mac**指令对应的MATCH和MASK宏定义：

汇编器需要定义一些宏来确定指令编码，其中有两个宏是一定要定义的，分别是MATCH和MASK，以**LUI**和**AUIPC**指令为例，



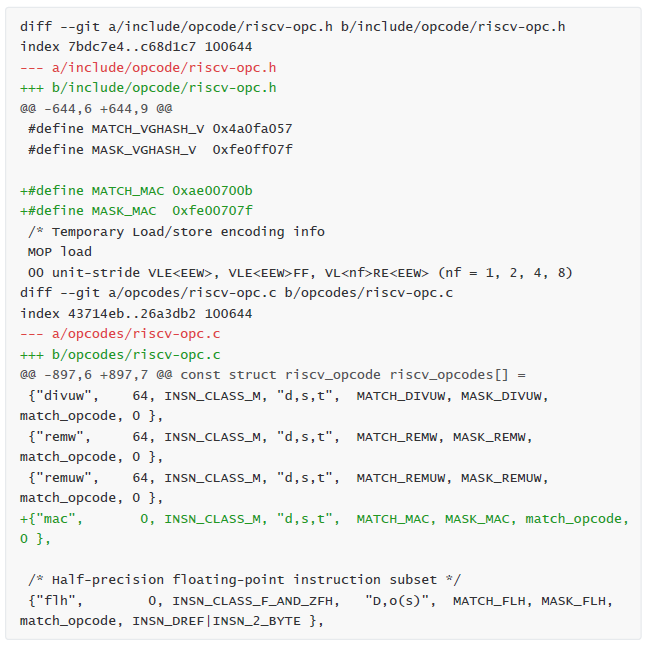
它们对应的MACTH和MASK宏为：



对于**LUI**和**AUIPC**指令，由于指令码的inst[31:7]是立即数，都是随机的，所以无法区分**LUI**和**AUIPC**指令，唯一能够区分两者的是inst[6:0]，所以MATCH\_LUI和MATCH\_AUIPC特定指明了两者的inst[6:0]编码，而MASK\_LUI和MASK\_AUIPC则是用于告知汇编器需要关注inst[6:0]编码，那么在生成指令码时，汇编器会这样使用这些宏定义：((insn ^ MATCH) & MASK) == 0. 如果不想手写这些宏定义的话，也可以借助github上的riscv-opcodes（https://github.com/riscv/riscv-opcodes.git）生成对应的MATCH和MASK宏定义。

* 1. 修改riscv-binutils 中的riscv-opc.c 和riscv-opc.h 文件：

将MATCH和MASK宏定义放在riscv-opc.h文件中，同时还需要在riscv-opc.c文件中描述**mac**指令。方便起见，我们暂且把**mac**指令归为M类指令。



1. **为gcc增加自定义指令**

为了让GCC识别**mac**指令，我们需要在GCC后端添加指令模板，由于**mac**指令可以描述成乘、加两种指令的混合体，所以我们可以直接使用GCC中自带的SPN来描述指令模板（对于AES、SHA这种加密指令，是无法直接使用GCC中自带的SPN来描述的，所以只能使用intrinsic，然后通过intrinsic与GCC后端的对应指令模板做匹配），经过修改后的riscv.md文件如下：



1. **在spike上增加自定义指令**

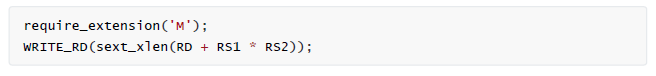
spike是一款RISC-V指令模拟器，我们可以在spike上增加自定义指令，运行包含自定义指令mac的应用程序，从而验证修改后的GNU工具链是否能够正常工作。

* 1. 更新riscv/encoding.h文件：

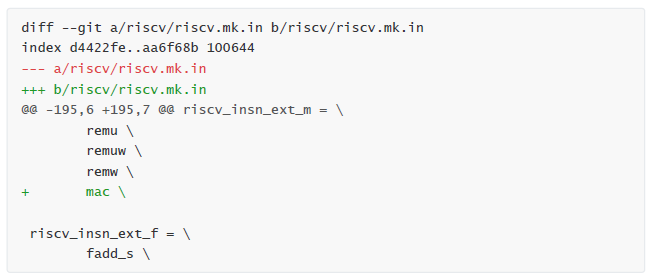


* 1. 增加指令描述：

在riscv/insns目录下新建文件mac.h，内容如下：



修改riscv/riscv.mk.in文件：



1. **构建、测试修改GNU工具链和spike**

在完成上面的修改后，需要重新构建GNU工具链和spike，在构建过程中，请确保您具有root权限。

* 1. 重新构建GNU工具链：

$ cd riscv-gnu-toolchain

$ mkdir build && cd build

$ ../configure --with-arch=rv64gc --with-abi=lp64d

$ make -j$(nproc)

* 1. 重新构建spike和pk：

$ cd riscv-isa-sim

$ mkdir build && cd build

$ ../configure

$ make -j$(nproc) && make install

$ cd riscv-pk

$ mkdir build && cd build

$ ../configure

$ make -j$(nproc) && make install

* 1. 测试

$ cat main.c

#include <stdio.h>

int32\_t op1 = 6;

int32\_t op2 = 7;

int32\_t res = 1;

int main(void)

{

res += op1 \* op2;

printf("res = %d\n", res);

return 0;

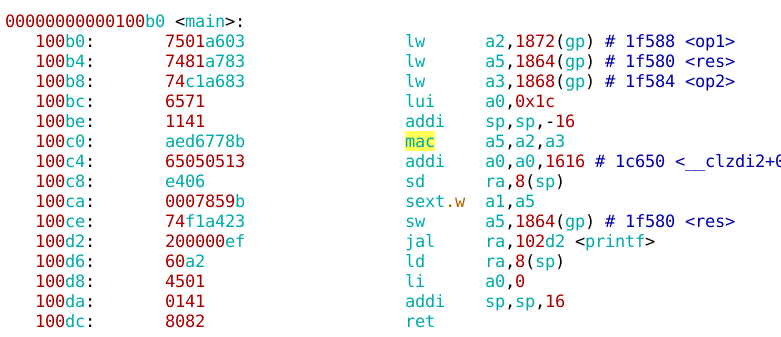
}

$ riscv64-unknown-elf-gcc –O2 main.c -o main.elf

$ riscv64-unknown-elf-objdump -D main.elf > main.asm

$ spike pk main.elf

查看反汇编文件main.asm：



1. **总结**

本文以在工具链中增加**mac**指令为例阐明整个开发流程，实质上只完成了其中最基础的部分。为了兼容标准的RISC-V指令集，我们还需要在GCC中增加如-march=rv64gc的编译选项，用于控制是否使用**mac**指令，同时为了表明应用程序中包含了**mac**指令，还需要在binutils中设置一些标志位，声明程序使用到了扩展指令**mac.**