1. 环境配置

**1.1**使用linux操作系统：WSL2或虚拟机（VMWare、VirtualBox等）。（Mac电脑可跳过。）

**WSL2安装使用（推荐）：** [window10通过Microsoft store安装ubuntu20.04 LTS](https://blog.csdn.net/yuanfen_330/article/details/122290451)

**虚拟机：**V**MWare、Virtu**alBox等。

**1.2**使用Docker

你可以访问 [Docker的官方网站](https://docs.docker.com/get-docker/) 来安装 Docker. 安装完毕后, 你可能需要重启你的系统.。

**1.3**获取编译实践的镜像

**1.3.1**

安装好docker后，在命令行输入：

docker pull maxxing/compiler-dev

过程比较漫长，耐心等待。

**1.3.2**

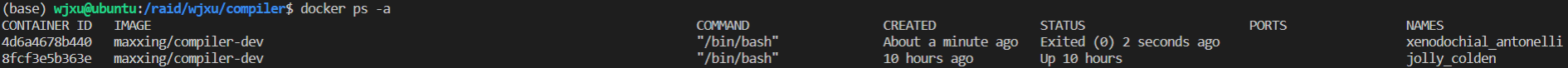
完成后再在命令行输入：

docker run -it maxxing/compiler-dev

就可以进入容器了：如下图



按ctrl+d就可以退出了。但这样做每次都会新增一条docker条目，退出docker使用docker ps -a会发现自己创建的docker：



建议记下CONTAINER ID和COMMAND字段使用以下命令：

docker exec -it 8fcf3e5b363e /bin/bash

这样每次进入和退出docker时不会增加新的条目，并且在docker中编写的代码和安装的软件包不会因为退出容器丢失。

**1.3.3**

将代码copy到docker中

如果使用wsl2的ubuntu的话，命令会比较简单：

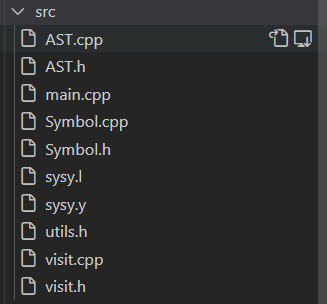
docker cp /mnt/e/sysy 8fcf3e5b363e:/root/

该命令将你电脑E盘下的sysy文件复制到docker中的root目录下。

1. 代码编写

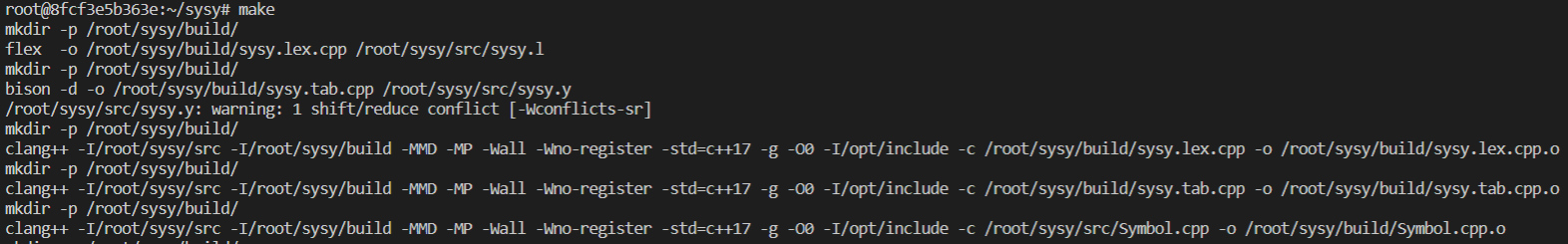
**2.1** sysy

在docker中进入sysy文件夹，你需要在src文件夹下编写你的编译器前端，中端和后端的代码：



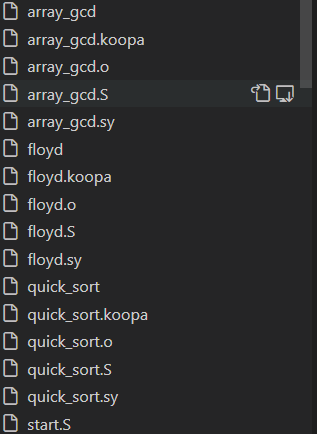
**2.2** 编译整个项目

使用make指令



**2.3** .elf文件

make后你将通过.sy文件得到文件.S .koopa .o和二进制文件：



1. 与CPU交互

**3.1** riscv-gnu-toolchain的使用

根据要求可以将该工具链编译成不同的工具，上面反汇编的例子中是riscv32-unknown-elf的objdump工具。

具体命令可参考GitHub源码：[riscv-gnu-toolchain源码](https://github.com/riscv-collab/riscv-gnu-toolchain)

也可参考其他教程：搜索riscv32-unknown-elf编译安装教程。

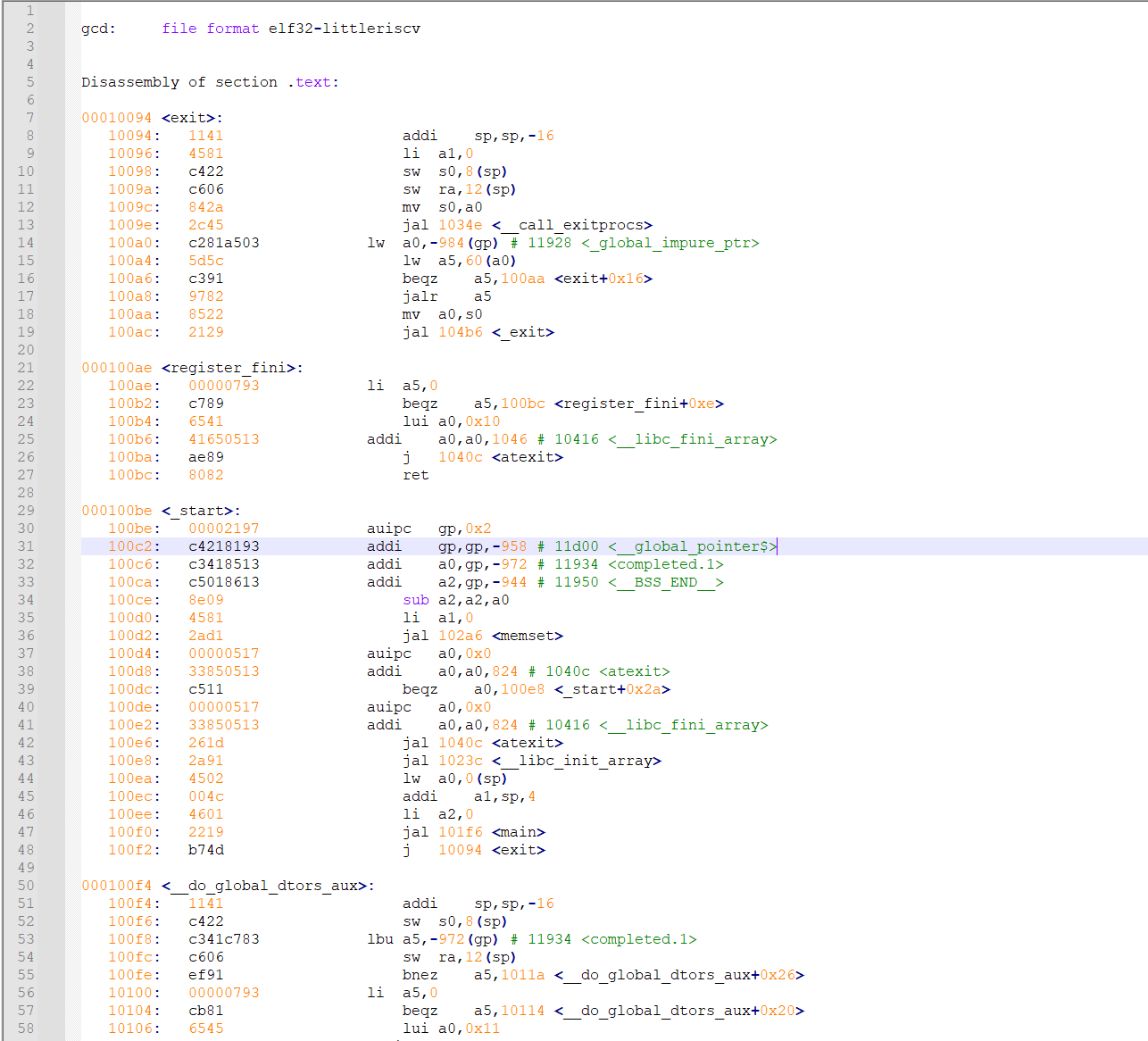
**3.2** 由二进制文件得到dump文件

可以使用riscv官方工具链riscv-gnu-toolchain中的riscv32-unknown-elf-objdump将上面得到的二进制文件行反汇编查看反汇编代码，来调试自己的cpu。

以wsl为例：



得到的quick\_sort.dump文件如下：



**3.3** 与CPU交互

正常情况下是要将对寄存器sp的赋值，数据段的初始化等等使用链接脚本和编写start.S文件来完成。但考虑到CPU，编译器和操作系统三项工作的同时进行，可以修改一下CPU硬件代码来配适编译器：

再者，由于FPGA板上的资源有限，32位的处理器的寻址空间达到2^32bytes，也就是4GB。这对于调试和FPGA板的烧写都是比较困难的。所以在设计CPU时可以考虑添加偏移量的方法。

以上面的quick\_sort.dump为例，程序从\_start段开始执行，pc为32’h100be。所以在初始化CPU时：

3.3.1.可以将pc的初始地址设置为32’h100be（正常该地址由操作系统决定）。

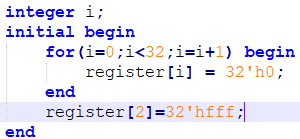


3.3.2.对于32位地址太宽的问题，可以在pc访问ins\_ROM的时候减去该段代码的偏移量：对于上面的quik\_sort.dump则是32’h10094。



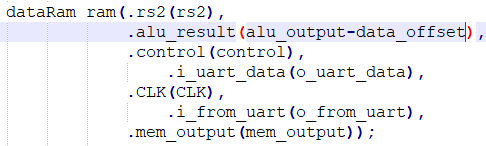


3.3.3.对于堆栈寄存器sp，可以在register file模块手动赋值。



3.3.4.还需要将上面dump文件的.data段的值赋值到dataRAM中，其中寻址也需要与上面ins\_ROM一致(减去偏移量)。





以上方法，仅适用于暂时没有办法将设计的编译器编译好的二进制代码放到设计的CPU上调试的情况，所以每次程序段的地址和程序大小改变后上面的内容都需要手动编写。上述方法还有个缺点就是对于程序的大小非常敏感。如果仅使用了地址的低16位如果寻址时地址超出该范围则会出bug。

参考资料

[北京大学编译实践课程在线文档 | 北大编译实践在线文档 (pku-minic.github.io)](https://pku-minic.github.io/online-doc/#/)

[北大2022编译原理实践（C/C++）-sysy 编译器构建 - lleozhang - 博客园 (cnblogs.com)](https://www.cnblogs.com/zhangleo/p/15963442.html)

北大编译实习课程公开测试用例

[riscv-gnu-toolchain源码](https://github.com/riscv-collab/riscv-gnu-toolchain)

[window10通过Microsoft store安装ubuntu20.04 LTS](https://blog.csdn.net/yuanfen_330/article/details/122290451)

[Docker的官方网站](https://docs.docker.com/get-docker/)