# 如何运行sodor的emulator

本文是关于ucb的The Sodor Processor的emulator的说明, 旨在能让读者成功运行sodor.写于2021年6月20日星期日. 有问题可以发送邮件到1263810658@qq.com

### 1.克隆仓库

ucb 的仓库https://github.com/ucb-bar/riscv-sodor融入chipyard中了, 因此没有emulator, 只有verilator.

我们需要

$git clone https://github.com/passlab/riscv-sodor.git --recursive

$ cd riscv-sodor

$ git submodule update --init –recursive

$ cd riscv-fesvr

$ ./configure --prefix=/usr/local

这一步是为了产生makefile文件.

$ make

这里可能有的错误:

Error during sbt execution: Error retrieving required libraries

(see /home/hadoop/.sbt/boot/update.log for complete log)

Error: Could not retrieve sbt 1.3.8

### 2. 下载sbt

首先查看自己的sbt环境. sbt sbt-version

可能会需要几分钟

getting org.scala-sbt sbt 0.13.0 (this may take some time)...他会尝试下载,但是由于sbt在国外所以大概率下载失败.

最后显示[error] [launcher] error during sbt launcher: error retrieving required libraries.

我们需要手动安装sbt. 网上许多手动安装sbt的教程我尝试后都是行不通的. 下面这个我可以安装成功.

第一步, 下载sbt压缩包

打开 <http://www.scala-sbt.org/download.html>, 下载最后的0.13.18版本比较保险. 因为接近需要的0.13.0环境. 下载下来后传到虚拟机里可以用vmware tool拖动, 服务器可以用lrzsz或者scp.这里不一一赘述.

第二步, 建立目录，解压文件到所建立目录

sudo tar zxvf sbt-0.13.5.tgz -C /opt/scala/

第三步, 建立启动sbt的脚本文件

/\*选定一个位置，建立启动sbt的脚本文本文件，如/opt/scala/sbt/ 目录下面新建文件名为sbt的文本文件\*/

$ cd /opt/scala/sbt/

$ vim sbt

/\*在sbt文本文件中添加

SBT\_OPTS="-Xms512M -Xmx1536M -Xss1M -XX:+CMSClassUnloadingEnabled -XX:MaxPermSize=256M"

java $SBT\_OPTS -jar /opt/scala/sbt/bin/sbt-launch.jar "$@"

然后按esc键 输入 :wq 保存退出，注意红色字体中的路径是定位到解压的sbt文件包中的sbt-launch.jar文件的绝对路径\*/

/×修改sbt文件权限×/

$ chmod u+x sbt

第四步 配置PATH环境变量，保证在控制台中可以使用sbt命令

$ vim ~/.bashrc

/\*在文件尾部添加如下代码后，保存退出\*/

export PATH=/opt/scala/sbt/:$PATH

/\*使配置文件立刻生效\*/

$ source ~/.bashrc

第五步, 设置源

vim ~/.sbt/repositories

[repositories]

local

huaweicloud-maven: https://repo.huaweicloud.com/repository/maven/

maven-central: https://repo1.maven.org/maven2/

huaweicloud-ivy: https://repo.huaweicloud.com/repository/ivy/, [organization]/[module]/(scala\_[scalaVersion]/)(sbt\_[sbtVersion]/)[revision]/[type]s/[artifact](-[classifier]).[ext]

第六步, 测试sbt是否安装成功

/\*第一次执行时，会下载一些文件包，然后才能正常使用，要确保联网了，安装成功后显示如下\*/

$ sbt sbt-version

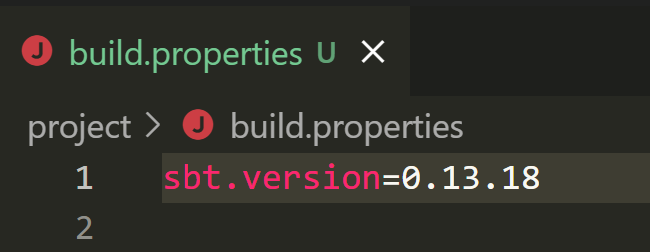
[info] Set current project to sbt (in build file:/opt/scala/sbt/)

[info] 0.13.5

### 3. 安装jdk

安装好sbt后再次make. 可能依旧make失败了

要修改 build.properties, 让他和我们安装的版本号一致.



error: scala.reflect.internal.MissingRequirementError: object java.lang.Object in compiler mirror not found.

因为需要jdk8

Jdk8可能需要oracle账号, 可以从镜像源下载, 然后传到虚拟机或服务器

<https://repo.huaweicloud.com/java/jdk/>

安装jdk这里不再赘述.

安装好后使用 alternatives 命令进行jdk版本的切换 1500 表示优先级

alternatives --install /usr/bin/java java /usr/local/java/jdk1.8.0\_162/bin/java 1500

alternatives --config java (会显示多个版本的jdk让你选择，直接根据数字进行切换)

java –version 如果显示1.8 那就可以了 java1.8就是java8

到这里应该make就可以成功了.

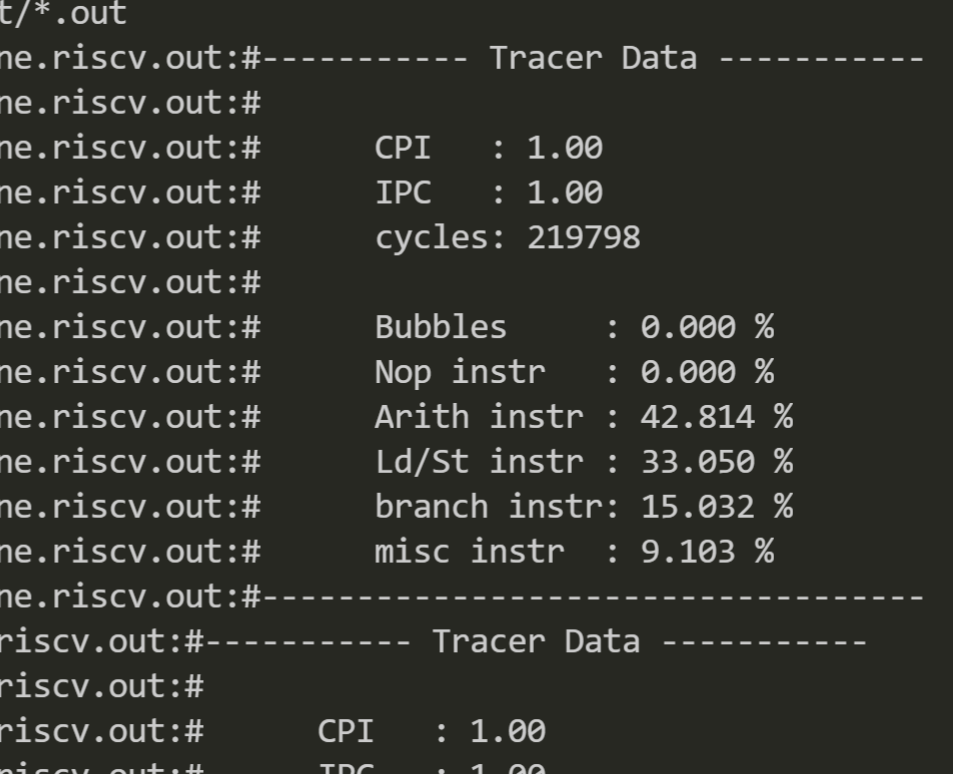
可以尝试

$ make run-emulator

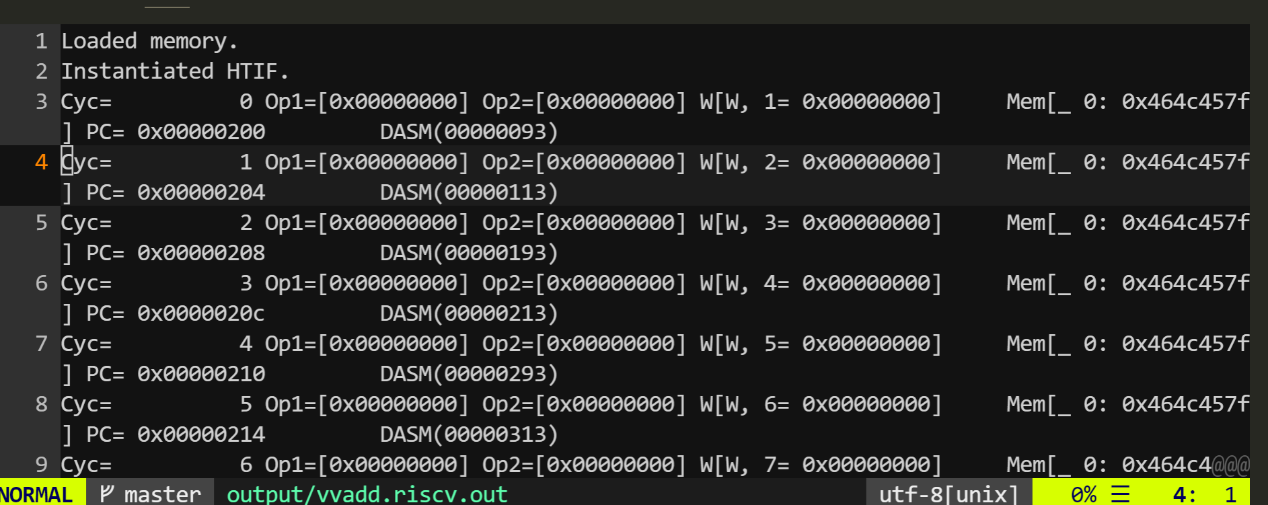
或者一个个run

cd emulator/rv32\_1stage 然后make run .

make report 可以看到



vim output/vvadd.riscv.out 可以看每个周期的指令在干嘛.



grep \# output/\*.riscv.out 选出重点.

下面附一个emulator的原理

make run 做了什么:  
运行 sbt，Scala 构建工具，选择 rv32 1stage 项目，并运行生成处理器的 Verilog RTL 描述的 Chisel 代码。 生成的 Verilog 代码可以在 emulator/rv32\_1stage/generated-src/ 中找到。  
l 运行verilator，一种将Verilog 编译为周期精确的C++ 仿真代码的开源工具  
l 将生成的C++ 代码编译为称为模拟器的二进制文件。   
l 运行仿真器二进制文件，将提供的 RISC-V 二进制文件加载到模拟内存中。 所有 RISC-V 测试和基准测试都将在调用 \make run 时执行。

如下图

