### Venus on Autolab

注:对于本课程,除Editor和Simulator外,其他功能非必须掌握。因此本指南相对侧重这两部分,其它未详尽说明的部分可参考其它资料:

Venus Reference | CS 61C Fall 2022 (berkeley.edu)

Home · ThaumicMekanism/venus Wiki · GitHub

以及本课程于2020年TA录制的Venus tutorial视频:

- https://robotics.shanghaitech.edu.cn/static/ca2020/CA2020 VenusTutorial1.mp4
- https://robotics.shanghaitech.edu.cn/static/ca2020/CA2020 VenusTutorial2.mp4
- https://robotics.shanghaitech.edu.cn/static/ca2020/CA2020 VenusTutorial3.mp4

# Venus

# 主界面

# **Terminal**

命令行文件系统,同学们可通过 help 指令自行了解。

### **Files**

文件系统图形界面,但相比Terminal功能不全,比如创建新文件须通过Terminal的 touch 命令。

# **URL**

生成一个URL,通过其打开的Venus某些部分会立即被初始化。

#### Auto set save

初始化Settings-Save on Close设置。

#### Auto set the code

初始化Editor中的代码为现在Editor中的。

选中某些设置项后,下方会自动生成URL。

还有些部分需要手动生成,详情参考Wiki。

# Wiki

内容不够完善。Venus中RISC-V的语法相关问题可在这里查到(如 .text 等Assembler directives, ecall 等)。

# JVM

Venus的Java版本,功能与网页版大致相同。

点击链接下载, (需Java环境) 命令行输入 java -jar venus-jvm-latest.jar 打开。

一般为助教使用,同学们可自行了解。

# **Settings**

# General

### **Simulator Default Args**

设置程序参数,形如 a . exe -b c 中的 -b c 。在C语言中,用 int argc,char\*\* argv 来接收参数;在 Venus中也相同,寄存器 a0 对应 argc , a1 对应 argv 。

#### **Text Start**

设置text segment的起始地址。

#### **Max History**

限制Simulator-Prev的最大回撤次数。默认为-1也就是不限制。

### **Save on Close**

默认不选中此项,关闭Venus页面,下次打开时文件系统、Settings等都将被重置!如选中此项,关闭 Venus时会保存当前状态。

### **Force Aligned Addressing?**

如选中此项,则强制要求内存内容按4字节对齐(当 lw , sw 等指令的目标内存地址非4的倍数时,程序报错并中止)。

#### **Mutable Text?**

如不选中此项,则禁止修改内存中的机器码(当发现修改时,程序报错并中止)。

### **Only Ecall Exit?**

是否必须调用 ecall 退出程序。默认不选中此项,则当PC超出代码范围时也会退出程序。如选中此项,则PC可能超出代码范围并引发问题。

当Simulator运行结束时,会弹出提示框 Exited with error code。若程序是因PC超出代码范围而退出的,则error code为寄存器 a0 的值;否则按照 ecall 的标准。

### **Set Registers on Init?**

如不选中此项,则程序开始时所有寄存器的值均为0。

### **Allow Access Between Stack and Heap?**

默认不选中此项(当 1w, sw 等指令的目标内存地址在stack和heap之间时,程序报错并中止)。

### Max number of steps

限制Simulator的最大运行步数(当超过时,程序报错并中止)。默认为-1也就是不限制。

### **Dark Mode**

亮/暗主题切换。

# **Calling Convention**

# **Enable Calling Convention Checker?**

启用Calling Convention检查,对于违反Calling Convention的行为输出警告信息。

### **Tracer**

### **Registers Pattern**

每一步输出的格式串,按上面说明的规则转义后输出。你可以修改,自定义一个格式。

#### **Instruction first?**

是否先读取指令。可以理解为Trace的时间节点选择,instruction first是在读取指令后、更新PC前;默认是在更新PC后、读取指令前。

#### **PC Word Addressed?**

是否按word表示PC(相比默认按byte表示,除以4)。

#### Two Stage?

是否模拟2-stage pipeline。

#### **Total number of commands**

设置固定trace多少次。默认为-1也就是到程序结束。

# **Output Number's Base**

设置以几进制输出数字。默认为2。

# **Packages**

选择是否启用Chocopy, Tester, Decoder这些工具。启用的才会出现在上方。

# **Editor**

在这里编辑汇编代码。

支持代码高亮, 支持复制粘贴等常见快捷键。

#### **Active File**

在页面最下方。如果正在编辑临时内容,则显示为null;否则说明正在编辑文件系统中的文件,显示为该文件的绝对路径,此时点**Clear**切换至编辑临时内容。

对于快捷键Ctrl+S,如果正在编辑临时内容,则待弹出提示框后输入文件名,将文件下载到本地;如果正在编辑文件,则保存。

□ 是否涉及RISC-V语法?

# **Simulator**

# 控制按钮区

### **Assemble & Simulate from Editor**

将Editor中的代码汇编并载入到Simulator中。若汇编成功则可以开始模拟,若汇编失败会弹出报错提示。

#### Cancel

不载入,保持Simulator现有状态。



#### Run

持续运行代码,直到断点处或程序结束。运行途中此按钮会显示转圈图案,此时点击按钮会中止运行。

### Step

下一步。

Prev

上一步。

#### Reset

重新开始。注意断点会被清除。

### **Dump**

导出机器码,显示在下方输出区。

#### **Trace**

从头到尾执行一次程序,每运行一步就输出一次调试信息,显示在下方输出区。详情见<u>Settings-</u> <u>Tracer</u>。

# 代码区

PC	Machine Code	Basic Code	Original Code
0x0	0x000002B3	add x5 x0 x0	main: add t0, x0, x0
0x4	0x00100313	addi x6 x0 1	addi t1, x0, 1
0x8	0x10000E17	auipc x28 65536	la t3, n
0xc	0x008E0E13	addi x28 x28 8	la t3, n
0x10	0x000E2E03	lw x28 0(x28)	lw t3, 0(t3)
0x14	0x000E0C63	beq x28 x0 24	fib: beq t3, x0, finish

四列依次显示PC,机器码,经过处理的汇编码,编辑器中原始汇编码。注意Basic Code和Original Code并不是——对应的,因为存在伪指令,可能对应地翻译为两条基本指令,例如上图中 Ta t3 n。

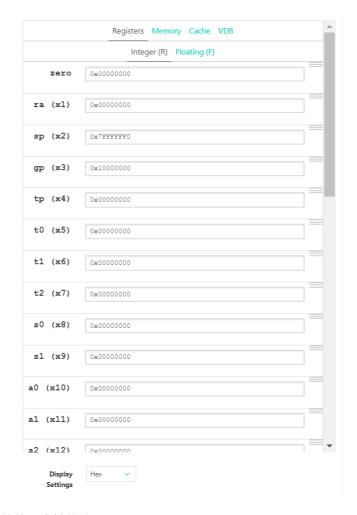
绿色显示将要运行的下一行,红色显示断点所在行。在某一行上点击即可添加或删除断点。

# 输出区

程序调用 ecall 的输出、调试信息、错误信息等都会显示在这个不可编辑文本框里。内容支持复制、下载、清除。

# 监视区

# Register



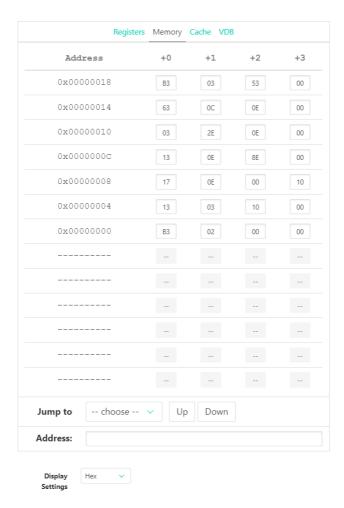
实时显示所有寄存器的值, 支持修改。

按住右侧三横线上下拖动,可调整寄存器的显示顺序。

# **Display Settings**

设置数值的显示格式,可选择十六进制、有符号十进制、无符号十进制、ASCII字符。

# **Memory**



实时显示一段内存的值, 支持修改。

# Jump to

跳转到特定位置显示, 有四个可选项。

# Up/Down

向上/向下移动视图。

### **Address**

跳转到自定义位置。在文本框输入地址数值,再点击文本框以外的地方才会更新视图。

# **Display Settings\***

同Register-Display Settings。

# Cache

	Registers Memory Cache VDB
Cache Levels	1
Block Size (Bytes)	4
Number of Blocks	1
Associativity	1
Cache Size (Bytes)	4
	Enables current selected level of the cache.
Direct Mappe	d
LRU ~	L1 V
Hit Count	0
Accesses	0
Hit Rate	???
0) EMPTY	
NOTE: This is	s a write through, write allocate cache.
Seed	4668807428150419571

留到完成Cache相关的练习时进行探索。

注意,完成设置后,点击Enable?才会启用Cache!

# **VDB**

	Registers Memory Cache VDB		
	Breakpoints Watchpoints		
Read/Write	On Read 🗆   On Write 🗆		
Address			
Value			
Mask			
	Add Watchpoint		
vvv Watchpoints vvv			
Read/Write	On Read 🖾   On Write 🗹		
Address	0x7FFFFEC		
Value	None		
Mask	None		
	Remove		

# **Breakpoints**

断点,程序运行到断点位置暂停,通过点击代码某一行添加或删除。无设置项。

# Watchpoints

监视点,程序发生满足条件的内存读写时暂停。4个设置项:

- Read/Write 选择在什么时候暂停(读、写、读或写)。至少选择一项。
- Address 指定监视内存地址。必填。
- Value 只当内存值等于填入值时候暂停。默认不填也就是不限制。

• Mask 地址掩码,若填入,则监视满足 x & Mask == Address & Mask 的所有地址 x 。可用于监视一个范围的内存地址。默认不填也就是监视单个地址。

设置完成后点击Add Watchpoint。

下方列出所有当前的Watchpoint,可修改设置项、删除。

# Chocopy

本课程无需使用,同学们可自行了解。

# **Tester**

一般为助教使用,同学们可自行了解。

# **Decoder (Disassembler)**

将十六进制机器码反汇编为RISC-V指令的工具,可能用得到。

#### Disassemble

(在Instruction Hex输入机器码后)点击反汇编。

**Use Registers Pseudo-names** 

设置是否使用伪寄存器名。

#### Add labels

设置是否添加label来取代跳转指令中的数字位移量。

**Decode to Pseudo Instruction if possible (experimental)** 

尝试进一步解译为伪指令。

Convert any line which does nothing to a nop

将实质无用的指令解译为 nop 。

**Instruction Hex** 

先在这里输入机器码。

**Disassembled Instructions** 

得到反汇编结果。

# **VS Code RISC-V Extensions**

助教使用过。通过安装以下两个插件,配置一个本地RISC-V环境。

# **RISC-V Support**

用来支持语法高亮。

# **RISC-V Venus Simulator**

支持Venus Simulator的大部分功能和一些其他功能。 打开.s文件,按F5调试,即启动Simulator。 详情参考插件的使用文档。