1

我是参考右书。

2

3

不知道大家是否想过这样一个问题，编译器有开源的gcc，操作系统有开源的Linux，那有没有开源的指令集呢？为什么不能有呢？

如果有一个会怎么样？

4

然后他们就启动了一个三个月的项目，四年之后终于发布了。

5

这里不展开讨论了，总之，他们需要修改指令集架构以及处理器核心

最后他们选择了RISC-V

http://prof.ict.ac.cn/bpoe_8/wp-content/uploads/baoyungang.pdf

http://acs.ict.ac.cn/baoyg/downloads/asplos2015_pard.pdf

计算所的案例作为一个例子。

6

今天我就给大家分享一下我去年学习的RISC-V，希望本次分享结束后大家能够为RISC-V写一个操作系统出来。我假设大家都已精通x86。

7

RISC-V最与众不同的几点，主要就是它简单，指令集规范只有大约200页，不像x86和ARM有几千页的文档；以及它没有历史包袱。熟悉x86的应该听说过20号地址线（也就是A20）以及段描述符中被拆散的那些字段。

另外，RISC-V的指令集是分模块的，之后会详细说明。

下面就让我们看一看RISC-V指令集的具体内容，看看他到底有没有说的这么好。

8

RISC-V指令集规范分为三卷：用户态指令集，最新稳定版本是2.2，2.3的草案正在进行中；特权指令集，最新草案是1.10，1.11的草案正在进行中；以及调试支持，我不会，所以不讲。

首先介绍用户态指令集。

9

刚刚提到RISC-V的指令集是分模块的。所有RISC-V核都要实现最最基本的指令集，而其他的指令集模块作为扩展。

I代表最基本的整数指令集，M是乘法除法指令集，A是原子操作指令集，F是浮点指令集。为了描述方便，G代表通用，是IMAFD。

下面主要按照RV32来介绍。首先介绍最基础的整数指令集。

10

11

RV有充足的寄存器资源，一共31个可用的寄存器，还有一个硬连线到0的“寄存器”。

12

注意返回地址是在被调用的函数体内保存的。提问：x86的返回地址在哪保存的?

13

其实我觉得CSR（寄存器）相关操作的指令应该算是部分特权指令了，所以会放在特权指令的地方讲。

14

这些指令根据带不带立即数可以分为两种。每个不带立即数的指令都有一个目标寄存器和两个源寄存器，带立即数类型的指令有目标寄存器、一个源寄存器和一个立即数。

加法指令不产生异常。

15

加载高20位立即数。顺便一提，其他算术逻辑运算的立即数是12位有符号数。

注意addi立即数是有符号扩展的。

16

这个立即数也是高20位。

x86-64可以基于%rip寻址了。pc是当前地址，不是下一条。跳转目标比较近的话，可以直接用一条立即数长的跳转指令。

17

听说有的指令集还可以在运行时动态改变字节序。

RV的硬件可以支持非对齐内存访问，也可以不支持，让代码来模拟。为啥load区分有无符号，而store不区分?

18

没有延时槽！！！没有条件码，减少处理器的内部状态。

19

link的意思是保存返回地址。

20

比如loader加载完某个ELF文件，可能就需要执行一下fence.i。没有in或out指令。

21

获得乘积的高位和低位分为两条指令：但真的要重复两次乘法吗？同样，需要做两次除法吗？

RV除零甚至不产生异常，不像某些指令集，加法都会产生异常。

除零的结果是，商为全1，余数为被除数。

22

需要注意的是，RV的原子操作最短只能操作一个字，不支持字节或者半字的操作（x86好像是可以对一个字节进行操作的），我去年踩过这个坑。另外需要注意，这个原子操作需要地址是对齐的。

详细解释lr/sc。其实lr/sc就已经能够实现绝大部分同步互斥操作了，AMO更方便吧。

内存一致性模型是relaxed。所有的 RV32A 指令都有一个请求位(aq)和一个释放位(rl)。aq 被置位的原子指令保证其它线程在随后的内存访问中看到顺序的 AMO 操作;rl 被置位的原子指令保证其它线程在此之前看到顺序的原子操作。 If the aq bit is set, then no later memory operations in this RISC-V hart can be observed to take place before the AMO. Conversely, if the rl bit is set, then other RISC-V harts will not observe the AMO before memory accesses preceding the AMO in this RISC-V hart.

aq：在其他核看来，该AMO后面的操作不会提前发生。

rl：在其他核看来，该AMO前面的操作不会拖后发生。

23

可能会失败，所以一般是一个循环中使用它。

（这好像不是cas的语义啊。。。）

24

不会，快快讲。

25

刚刚提到的指令都是32位的，比较长，为了减小代码的体积，压缩扩展被提出。

Hardware designs can simply expand RVC instructions during decode, simplifying verification and minimizing modifications to existing microarchitectures.

Compilers can be unaware of the RVC extension and leave code compression to the assembler and linker, although a compression-aware compiler will generally be able to produce better results.

一条16位指令一定有一条32位指令对应，因此实现时只需要在指令译码器做改动即可。也因此，编译器可以无视RVC的存在，让汇编器和链接器去压缩代码。

这比x86好得多，听说ARM的Thumb也没有这一性质，因为Thumb甚至是一个新的ISA。

26

27

28

那本书还给了个插入排序的汇编程序，感觉稍微有点硬核。

首先把栈拉下来，然后把返回地址存到栈上，然后通过两条指令得到格式化字符串的地址作为printf的第一个参数，然后同样获得string2的地址作为第二个参数。这里是绝对地址，而不是pc相对寻址。然后调用printf。然后把返回地址从栈中取回来，return 0。

29

原来RV32的指令都是对32位寄存器操作的，这里变成64位即可。

注意，每条指令本身的长度保持为四字节（32位）或两字节（16位）不变。所以指令中立即数的长度不变，改变的是寄存器的长度。

这是我自己总结的，可能有错。

30

31

我个人理解，对于定长指令集而言，加载长立即数不是一件平凡的事情。

注意，RV64的lui指令将立即数加载到寄存器的12位到31位，并进行符号扩展。

32

33

34

这是我自己总结的，可能有错。

35

讲完用户态后，讲特权指令。这里是按照1.10版本讲的。新版可能发生变化。

36

37

RV的特权级分为四层。如上图所示。

提问，猜一下第二级可以用来做什么。

有的说法是，第二级特权级是留给虚拟化的hypervisor的。

这些特权级都是可选的。对于不同的使用场景，某个芯片可以选择只实现M，或者实现M、U，或者实现MSU等。如下图所示。

38

对处理器进行配置的本质就是寄存器的设置。

39

前四个寄存器和misa描述了这个处理器的信息。比如支持哪些扩展指令集以及处理器号。

40

41

操作CSR的指令就这么简单，但是为了配置好CSR，还是不容易的。

42

下面开始具体讲每个CSR，虚拟内存机制，以及中断和异常的处理会穿插讲解。

SPIE和SPP是发生异常和中断之前的IE和特权级。x86这些信息存在哪？中断发生时处理器会压栈。RV异常和中断的处理过程在之后讲。

关于SUM，1.10允许执行用户的代码，1.11更新为：The virtual-memory system **no longer** permits supervisor mode to execute instructions from user pages, regardless of the SUM setting.

43

页表基址，类似x86的cr3

44

页表基址，类似x86的cr3

45

这里以Sv39为例子。

46

RISC-V规定A、D位可以由硬件在相应时机置位，也可以不置位而是产生异常由软件处理。x86则是硬件负责置位。提问：何时置位，谁负责清零，何时清零。

当RWX都为0时代表还有下一级页表，否则代表这就是一个叶子节点，一个页了。RWX都为0可以出现在各个页表层级，分别支持4K、2M、1G的普通页、大页和巨大页。

47

这是个Sv32的图。

48

我看看我的备注怎么写的...哦，没想让大家看清，放在这装装样子。

49

50

sie表示是否开启（不屏蔽）软件中断、定时器中断和外部中断，sip表示是否有相应中断未处理。

51

x86是如何配置中断向量表的？IDT。

52

x86中sepc这些信息存在哪？中断发生时处理器会压栈。

发生异常时的pc，以及cause是异常号。

53

page fault的虚拟地址，类似x86的cr2

54

这个寄存器存啥都行。

可以仔细讨论一下如何换栈，并与x86对比。

55

通过设置这两个寄存器，可以让硬件直接把中断或异常发给S态而不是M态。

56

M态也有内存保护，但没有地址映射。

57

就这么几条，简单吧。

不过，其实我觉得CSR（寄存器）相关操作的指令应该算是某种程度的特权指令了。然而CSR（寄存器）有很多。

58

mret和sret用来从异常处理程序中返回，首先看看异常到来的时候如何处理。

首先保存一些必要信息，然后以相应特权级的身份跳到异常处理程序入口点（也就是异常处理向量）。

根据xtvec中的设置，也可能是pc = xtvec + 4 * cause

xcause和xtval也被相应设置。

59

从异常处理程序中返回，实际上就是处理一些寄存器，然后跳到mepc或sepc处。

xstatus.xPP和xstatus.xPIE会被设置为0和1，此处省略。

60

x86有个invlpg。

TLB shootdown: 1) a local data fence to ensure local writes are visible globally, then 2) an interprocessor interrupt to the other thread, then 3) a local SFENCE.VMA in the interrupt handler of the remote thread, and finally 4) signal back to originating thread that operation is complete.

61

有点像x86的hlt。

62

HiFive1 59美元，HiFive Unleashed 999美元，K210几十块钱人民币

HiFive Unleashed其实有5个核，有个不好用。。。

63

64

65

这几页说RISC-V形势一片大好。

66

67

68

RISC-V的目标。

69

70