PA1 实验报告

韩加瑞

2023年10月4日

1 必做题

1.1 程序是个状态机

```
(0, x, x)
(1,0,x)
(2,0,0)
(3,0,1)
                  the 1st loop
(4,1,1)
(3,1,2)
                  the 2nd loop
(4,3,2)
. . . . . .
(3,4851,99)
                  the 99th loop
(4,4950,99)
(3,4950,100) |
                  the 100th loog
(4,5050,100)
(5,5050,100)
```

1.2 理解基础设施

一个学期在调试上花费的时间为 500*90%*30*20/60/60=75h。 简易调试器可以节省 $\frac{2}{3}*75=50h$ 。

1 必做题 2

1.3 RTFM-riscv32

31	25	24 20	19	15 14 1	2 11	7 6	0
funct7	,	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]			rs1	funct3	$_{ m rd}$	opcode	I-type
imm[11:	:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
		imm[31:12]	rd	opcode	U-type		

图 1: base instruction formats

riscv32 有几种指令格式? 如图即为 riscv32 最基本的指令格式,此外还有立即数不同而产生的六种格式如下:

31 30 25	24 21 20	19	15 14 12	2 11 8	7	6 0	
funct7	rs2	rs1	funct3	rd		opcode	R-type
imm[1]	1:0]	rs1	funct3	rd		opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4	1:0]	opcode	S-type
imm[12] $imm[10:5]$	rs2	rs1	funct3	imm[4:1]	imm[11]	opcode	B-type
	imm[31:12]					opcode	U-type
imm[20] $imm[10]$	0:1] imm[11]	imm	[19:12]	$_{\mathrm{rd}}$		opcode	J-type

图 2: base instruction formats showing immediate variants

LUI 指令的行为是什么? LUI (load upper immediate) is used to build 32-bit constants and uses the U-type format. LUI places the U-immediate value in the top 20 bits of the destination register rd, filling in the lowest 12 bits with zeros.

mstatus 寄存器的结构是什么样的? mstatus 寄存器的结构如下图所示。

1 必做题 3

The mstatus register is an MXLEN-bit read/write register formatted as shown in Figure 3.6 for RV32 and Figure 3.7 for RV64. The mstatus register keeps track of and controls the hart's current operating state. A restricted view of mstatus appears as the sstatus register in the S-level ISA.



Figure 3.6: Machine-mode status register (mstatus) for RV32.

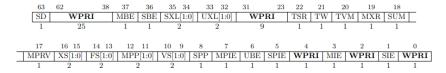


Figure 3.7: Machine-mode status register (mstatus) for RV64.

1.4 shell 命令

nemu/目录下总共有 24190 行代码, 使用的命令是:

find . | grep "\.[ch]\$" | xargs wc
$$-1$$

在 pa0 分支中的代码总量为 23801 行,我在 pa1 中总共编写了 381 行代码。除去空行之外,原本的 nemu/下总共有 20627 行代码,编写 pa1 后总共有 21000 行代码。

1.5 CFLAGS

-Wall 和 -Werror 可以在编译时刻把潜在的 fault 直接转变成 failure, 让编译器不放过任何一个错误。使用上述命令可以提前发现 bug。