

(1) 21:30:02







## P6\_L4\_sbc 文件上传

题目编号 939-976



## Sort depend on Bit Count

- 1	
ш	ы

-		4	n
н	L	л	
и			W



编码	31 26	25 21	20 16	15 11	10 6	5 0
	SPECIAL 000000	rs	rt	rd	0 00000	SBC 110001
	6	5	5	5	5	6
+42						

格 sbc rd, rs, rt 式

描述

将 GPR[rs] 每 8 位分成一组共 4 组,统计每组中 1 出现的个数。若 GPR[rt] 为奇数,则将这四组按照升序写入 GPR[rd](即高八位的 1 的个数最多,低八位的 1 的个数最少)否则按照降序写入 GPR[rd]。排序为稳定排序,即若两个部分的 1 的个数相等,则保持原来的顺序

 $bit_count(a[7:0]) = a[0] + a[1] + a[2] + a[3] + a[4] + a[5] + a[6] + a[7]$ 

condition ← GRF[rt] & 1

操 作 bit\_compare\_inc(a, b) {return bit\_count(a) < bit\_count(b)}

bit\_compare\_decc(a, b) {return bit\_count(a) > bit\_count(b)}

if condition GRF[rd]  $\leftarrow$  stable\_sort(GPR[rs], bit\_compare\_inc)

else GRF[rd] ← stable\_sort(GPR[rs], bit\_compare\_dec)

示例

sbc \$t0, \$t1, \$t2

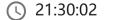
其 他 指令表现以 .class 指令扩展在 MARS 中的行为为准,对于 stable\_sort 函数的具体行为,请看下文实例

## stable\_sort

stable\_sort 接受两个参数,第一个参数是一个 32 位数,第二个参数是排序的方式,应当返回一个 32 位数

有如下示例











## 应当返回



ans = 32'h010307ff



这是因为由 bit\_compare\_dec 可知,要采用降序排列。待排列的四个数是



8'h07, 8'hff, 8'h03, 8'h01



它们二进制中 1 的个数分别为 3,8,2,1



```
8'h07 => 3 个 1
8'hff => 8 个 1
8'h03 => 2 个 1
```

8'h01 => 1 个 1

所以要将其排列为 ff, 07, 03, 01 这四个数, 最终一起输出, 即 32'h010307ff。 (似乎 和前面的顺序反转了,但是是因为书写的时候将低位写在后面,所以看上去是升序的,但是 实际是降序的)。

对于稳定排序,有如下解释

```
stable_sort(32'h50050306, bit_compare_inc);
```

8'h50, 8'h05,8'h03, 8'h06 每个数里都有 2 个 1 。所以它的输出就是输入顺序,不应该 发生变化, 即答案为 32'h50050306。

其注意,即使这里解释时近似于 C 语言函数的语法,但是我们并没有表示其实现方式一定是 用 Verilog 中的 function 或者 task 实现的。



SBC.CLASS



