

中国科学院大学计算机组成原理实验课

实 验 报 告

学号: 2015K8009929045 姓名: 张远航 专业: 计算机科学与技术

实验序号: 4 实验名称: 可执行程序的加载与硬件性能计数器

注 1: 本实验报告请以 PDF 格式提交。文件命名规则: [学号]-PRJ[实验序号]-RPT.pdf, 其中文件名字母大写, 后缀名小写。例如: [2014K8009959088]-PRJ[1]-RPT.pdf
注 2: 实验报告模板以下部分的内容供参考, 可包含但不限定如下条目内容。

一、 逻辑电路结构与仿真波形的截图及说明 (比如关键 RTL 代码段 {包含注释} 及其对应的逻辑电路结构、相应信号的仿真波形和信号变化的说明等)

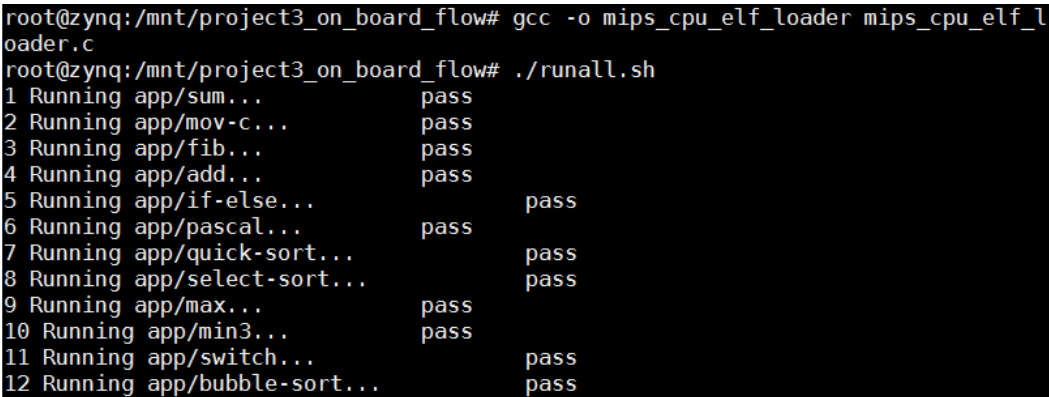
以下是 Loader 的关键代码段:

```
1      // TODO: fix the magic number with the correct one
2      const uint32_t elf_magic = 0x464c457f;
3      uint32_t *p_magic = (void *)buf;
4      // check the magic number
5      assert(*p_magic == elf_magic);
6
7      // our MIPS CPU can only reset with PC = 0
8      assert(elf->e_entry == 0);
9
10     for(i = 0, ph = (void *)buf + elf->e_phoff; i < elf->e_phnum
        ; i++) {
11         // scan the program header table, load each segment into
            memory
12         if(ph[i].p_type == PT_LOAD) {
13             uint32_t addr = ph[i].p_vaddr;
14
15             if(addr >= MIPS_CPU_REG_TOTAL_SIZE) {
16                 // Ignore segments with address out of ideal
                    memory
17                 // All segments we need to load can fit in the
                    ideal memory
18                 continue;
19             }
20
```

```

21         // TODO: read the content of the segment from the
           ELF file
22         // to the memory region [VirtAddr, VirtAddr +
           FileSiz)
23         // Use file operations
24         // Use `mips_addr(addr)` to refer to address in mips
           CPU
25         Elf32_Word len = ph[i].p_filesz;
26         int rd_success = fseek(fp, ph[i].p_offset, SEEK_SET)
           ;
27         assert(rd_success == 0);
28
29         Elf32_Addr *paddr = (Elf32_Addr*)mips_addr(addr);
30         fread(paddr, sizeof(uint8_t), len, fp);
31         // TODO: zero the memory region
32         // [VirtAddr + FileSiz, VirtAddr + MemSiz)
33         len = ph[i].p_memsz - ph[i].p_filesz;
34         paddr += ph[i].p_filesz;
35         memset(paddr, 0, len);
36     }
37 }

```



```

root@zynq:/mnt/project3_on_board_flow# gcc -o mips_cpu_elf_loader mips_cpu_elf_l
oader.c
root@zynq:/mnt/project3_on_board_flow# ./runall.sh
1 Running app/sum...      pass
2 Running app/mov-c...    pass
3 Running app/fib...      pass
4 Running app/add...      pass
5 Running app/if-else...  pass
6 Running app/pascal...   pass
7 Running app/quick-sort... pass
8 Running app/select-sort... pass
9 Running app/max...      pass
10 Running app/min3...     pass
11 Running app/switch...   pass
12 Running app/bubble-sort... pass

```

loader 上板运行通过的截图

性能计数器的相应代码段如下（要求的输出信号是在数据通路模块中通过实例化产生的），这一部分添加在了控制单元中：

```

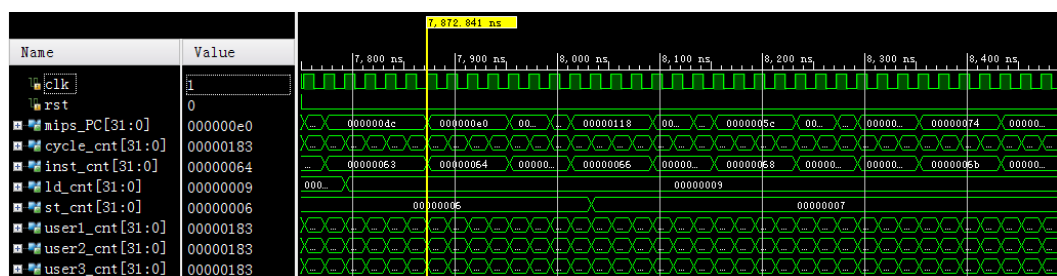
1  // perf. counter
2      always @(posedge clk)
3          if (rst) begin

```

```

4         mips_perfcntr_br <= 32'd0;
5         mips_perfcntr_ld <= 32'd0;
6         mips_perfcntr_st <= 32'd0;
7         mips_perfcntr_cycle <= 32'd0;
8         mips_perfcntr_inst <= 32'd0;
9         mips_perfcntr_user1 <= 32'd0;
10        mips_perfcntr_user2 <= 32'd0;
11        mips_perfcntr_user3 <= 32'd0;
12    end
13    else begin
14        mips_perfcntr_cycle <= mips_perfcntr_cycle + 1;
15        if (state == FETCH) mips_perfcntr_inst <=
            mips_perfcntr_inst + 1;
16        if (state == DECODE) begin
17            case (Op)
18                SPECIAL: if (func == JR) mips_perfcntr_br <=
                    mips_perfcntr_br + 1;
19                LW: mips_perfcntr_ld <= mips_perfcntr_ld +
                    1;
20                SW: mips_perfcntr_st <= mips_perfcntr_st +
                    1;
21                J, JAL, BEQ, BNE: mips_perfcntr_br <=
                    mips_perfcntr_br + 1;
22            endcase
23        end
24        mips_perfcntr_user1 <= mips_perfcntr_user1 + 1;
25        mips_perfcntr_user2 <= mips_perfcntr_user2 + 1;
26        mips_perfcntr_user3 <= mips_perfcntr_user3 + 1;
27    end

```



运行 quick_sort 过程中各性能计数器的输出

二、 实验过程中遇到的问题、对问题的思考过程及解决方法（比如 RTL 代码中出现的逻辑 bug，仿真及上板调试过程中的难点等）

- 压缩包没更新之前，`cycle_cnt` 等计数器都被优化掉了，后来更新了压缩包就好了。
- 现象：用 Post-Implementation 做仿真时，PC 的值有些位一直是 Z（高阻态）。关闭 `flatten_hierarchy` 选项之后，在 Post-Implementation Simulation 中能找到一个显示正常的 PC 信号 `^mips_PC[31:0]`。后来经过思考，按照我配置的 `ideal_memory`，PC 的高位没用上，而内存又是四字节对齐的，低两位也用不上，所以自然都被优化掉了，出现高阻态也正常。（但是为什么上一次实验就没出现这种情况呢？）
- 一开始计数器的代码分散在各个 `always` 块里，看着让人相当不爽，就给拎出来单独放到一个 `always` 块了。

三、 对讲义中思考题的理解和回答

本次实验无思考题

四、 对于此次实验的心得、感受和建议（比如实验是否过于简单或复杂，是否缺少了某些你认为重要的信息或参考资料，以及其他想与任课老师交流的内容等）

这次实验真的好简单，没什么想说的……感觉可以整合到别的实验里。但是助教哥哥讲的关于程序编译到执行的那些知识是无价的！