Lab3-2-exam

创建并切换分支

```
git checkout lab3
git add .
git commit -m "xxxxx"
git checkout -b lab3-2-exam
```

题目描述

本次题目我们将基于时间片轮转实现一种新的调度方式。

调度规则

- 在全局有三个调度队列，记为 `env_sched_list[3]` ，每个队列中的进程单次运行的时间片数量为进程优先级乘以不同的权重，具体的：
  - `env_sched_list[0]` 中进程单次运行时间片数 = 进程优先级数 * **1**
  - `env_sched_list[1]` 中进程单次运行时间片数 = 进程优先级数 * **2**
  - `env_sched_list[2]` 中进程单次运行时间片数 = 进程优先级数 * **4**
- 进程创建时全部插入到第一个调度队列（即 `env_sched_list[0]` ）的**队首**
- 进程时间片用完后，根据自身优先级数值加入到另外两个调度队列**队尾**，若自身优先级为**奇数**，则顺次加入到**下一个调度队列**队尾，若为**偶数**，则加入到**下下个调度队列**队尾。当然，进程不再处于原调度队列中。具体地：
  - `env_sched_list[0]` 中的进程时间片耗完后，若优先级为奇数，加入到 `env_sched_list[1]` 队尾；若优先级为偶数，加入到 `env_sched_list[2]` 队尾
  - `env_sched_list[1]` 中的进程时间片耗完后，若优先级为奇数，加入到 `env_sched_list[2]` 队尾；若优先级为偶数，加入到 `env_sched_list[0]` 队尾
  - `env_sched_list[2]` 中的进程时间片耗完后，若优先级为奇数，加入到 `env_sched_list[0]` 队尾；若优先级为偶数，加入到 `env_sched_list[1]` 队尾
- `sched_yield` 函数首先从 `env_sched_list[0]` 队列开始调度，之后依次按照 `0`, `1`, `2`, `0`, `1`, ……的顺序切换队列，且**仅在当前队列中没有可运行进程时切换到下一个队列**
- **其他关于进程是否可以运行的条件与 lab3 课下要求相同**

题目要求

- 修改操作系统相应文件，使得进程调度队列个数为 **3**
- 修改 lib/sched.c 文件中的 `sched_yield` 函数，实现上述调度规则

  为便于评测，请在每次调用 `sched_yield` 函数时**打印换行符**，将连续运行的两进程输出分隔开：

  ```
  printf("\n");
  ```

注意

- 若运行时发现程序进入预期之外的异常，可能是由于编译器生成了 R3000 不支持的汇编指令，请修改 `学号` 目录下的 `include.mk` 文件，在 `CFLAGS` 后面加上 `-mips2` 参数，如果有 `-march=r3000` 参数，也需要去掉。

  修改后再编译运行，若仍无法解决，请检查自己的代码实现，也可到MOOC讨论区 `lab3-2-exam` 主题下提问。
- 测试中使用的进程不完全来自 code_a.c 和 code_b.c，若无法通过测试，你可能仍需检查 ELF 加载过程。
- 评测保证**至少有一个 RUNNABLE 的进程。**
- 请务必在提交前注释掉所有新增的用于调试的 printf 输出，仅保留题目要求输出的换行符，否则可能影响评测。

本地测试

将 init/init.c 中的 `mips_init` 函数替换为如下内容：

```
void mips_init()
{
    printf("init.c:\tmips_init() is called\n");
    mips_detect_memory();

    mips_vm_init();
    page_init();

    env_init();

    // for lab3-2-exam local test
    ENV_CREATE_PRIORITY(user_A, 2);
    ENV_CREATE_PRIORITY(user_B, 1);

    trap_init();
    kclock_init();
    panic("^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^");
    while(1);
    panic("init.c:\tend of mips_init() reached!");
}
```

运行如下指令：

```
make clean && make && /OSLAB/gxemul -E testmips -C R3000 -M 64 gxemul/vmlinux
```

观察输出（下面的省略号不是输出，与示例不完全相同，对照方法见下）：

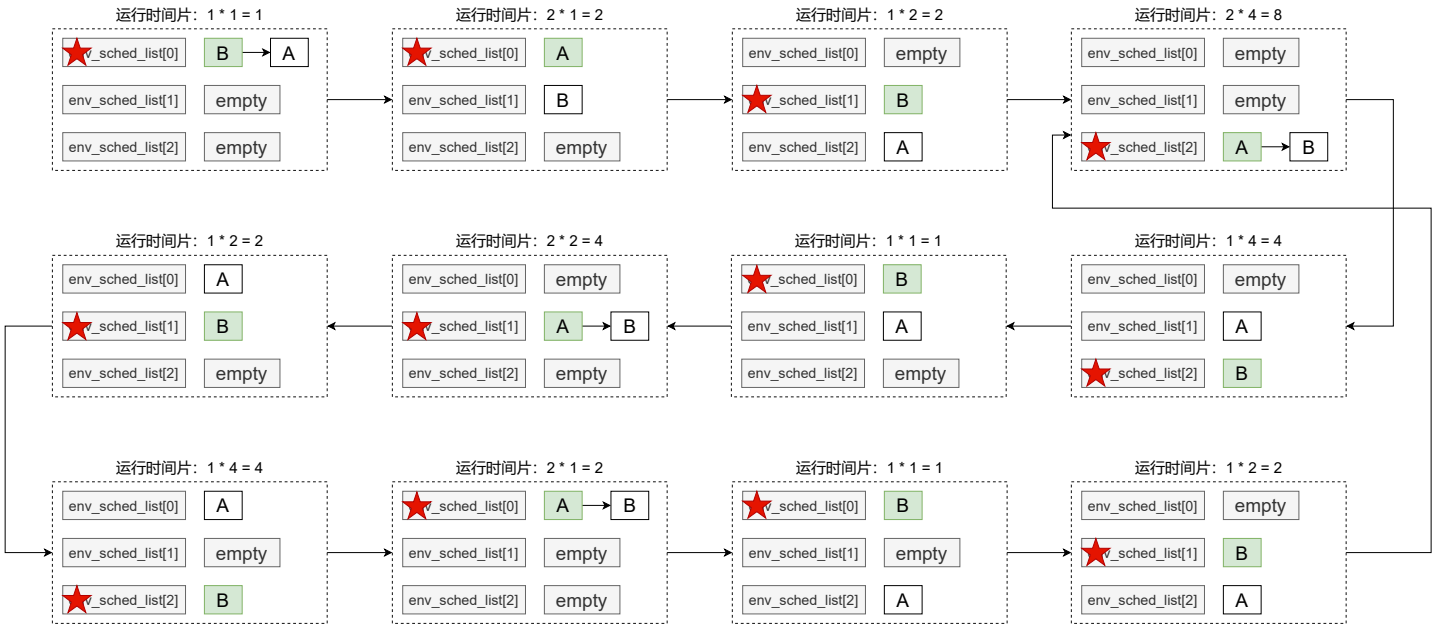
```
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    .......
```

显示结果中每一段表示一次进程输出，其中数字个数不尽相同，但测试只关注进程调度顺序，以及每种数字输出的行数。具体地，在本样例中，从操作系统初始化结束开始，首先 1 行 2 ，2 行 1， 2 行 2，之后按如下顺序及组成循环：

- 8 行 1
- 5 行 2
- 4 行 1
- 6 行 2
- 2 行 1
- 3 行 2

给出本样例的调度图解如下，其中绿色和白色方框中的字母 A 和 B 分别表示使用 user_A 和 user_B 创建的进程（具体地，user_A 进程打印 1，优先级为2；user_B 进程打印 2，优先级为 1），星形标识的为正在调度的队列，绿色方框表示正在运行的进程：



代码提交

```
git add .
git commit -m "xxxxx"
git push origin lab3-2-exam:lab3-2-exam
```

Lab3-2-Extra

创建并切换分支

```
git checkout lab3
git add .
git commit -m "xxxxx"
git checkout -b lab3-2-Extra
```

问题描述

在本次 Extra 中，我们希望大家针对地址错误中的 `AdEL` 错误进行处理。

这个异常的触发有两种情况：一种是在用户态试图**读取** kuseg 外的地址，另一种是试图从一个不对齐的地址读取字或半字（如 `lw $t0, 1($0)` 就会触发该异常）。

我们希望大家针对上述的第二种情况进行异常处理，处理方式如下：

- 如果异常由 `lw` 指令触发，则将发生异常的指令替换为 `lh` 指令。
- 如果异常由 `lh` 指令触发，则将发生异常的指令替换为 `lb` 指令。

指令替换过程**只修改指令 26-31 位**，不修改寄存器编号和 offset 字段。

涉及到的指令格式如下：

opcode	rs	rt	immediate
31 - 26	25 - 21	20 - 16	15 - 0

各指令 opcode 值如下：

指令	opcode
lb	100000
lh	100001
lw	100011

提示

1. 请阅读 *See MIPS Run Linux* 的第 58 页（英文版第 66 页），找到 AdEL 指令的异常号。
   注：上述所述第 *x* 页指的是书的页码，而不是 *PDF* 的页码，文档可以在*MOOC*下载
2. 请在完成异常处理函数后修改 lib/traps.c 中的 `trap_init` 函数，将你自己编写的异常处理函数加入异常向量组中的对应位置。
3. 大家可以使用 lib/genex.S 中定义的 `BUILD_HANDLER` 宏来构建自己的异常处理函数，构建方法可以参考已有的 `handle_tlb` 等处理函数。

本地测试

为了进行本地测试，请在 **init/** 目录下创建 **test.S** 文件，并填入以下内容：

```
#include <asm/regdef.h>
#include <asm/cp0regdef.h>
#include <asm/asm.h>
#include <stackframe.h>

LEAF(test1)
        addu a0, a0, a1
        lw v0, 0(a0)
        jr ra
END(test1)

LEAF(test2)
        addu a0, a0, a1
        lw v0, 0(a0)
        jr ra
END(test2)
```

并将该目录下的 Makefile 文件改为如下内容，注意**若直接从下发的Makefile.txt文件中复制，请保证缩进为一个Tab而非多个空格。**

```
INCLUDES := -I../include

%.o: %.c
        $(CC) $(CFLAGS) $(INCLUDES) -c $<

%.o: %.S
        $(CC) $(CFLAGS) $(INCLUDES) -c $<

.PHONY: clean

all: init.o main.o code_a.o code_b.o check_icode.o test.o

clean:
        rm -rf *~ *.o


include ../include.mk
```

之后将 **lib/kclock_asm.S** 中的 `setup_c0_status STATUS_CU0|0x1001 0` 这一行注释掉；

最后在 **init/init.c** 中添加测试函数并调用：

```
extern u_int test1(char *p, u_int offset);
extern u_int test2(char *p, u_int offset);

void test() {
        char a[100] = {5, 4, 3, 2, 1};
        int i = 0;
        // lw -> lh
        i = test1(a, 2);
```

```
        printf("%08x\n", i);
        // lh -> lb
        i = test1(a, 3);
        printf("%08x\n", i);
        // lw -> lb
        i = test2(a, 1);
        printf("%08x\n", i);
}

void mips_init()
{
        printf("init.c:\tmips_init() is called\n");
        mips_detect_memory();

        mips_vm_init();
        page_init();

        env_init();

        trap_init();
        kclock_init();
        test();
        *((volatile char *) 0xB0000010);
}
```

运行如下指令：

```
make clean && make && /OSLAB/gxemul -E testmips -C R3000 -M 64 gxemul/vmlinux
```

正确的输出为：

```
main.c: main is start ...

init.c: mips_init() is called

Physical memory: 65536K available, base = 65536K, extended = 0K

to memory 80401000 for struct page directory.

to memory 80431000 for struct Pages.

pmap.c:  mips vm init success

00000302

00000002

00000004
```

代码提交

```
git add .
git commit -m "xxxxx"
git push origin lab3-2-Extra:lab3-2-Extra
```