

“龙芯杯”第四届全国大学生计算机系统能力培养大赛报告

复旦大学计算机学院FDU1.2队

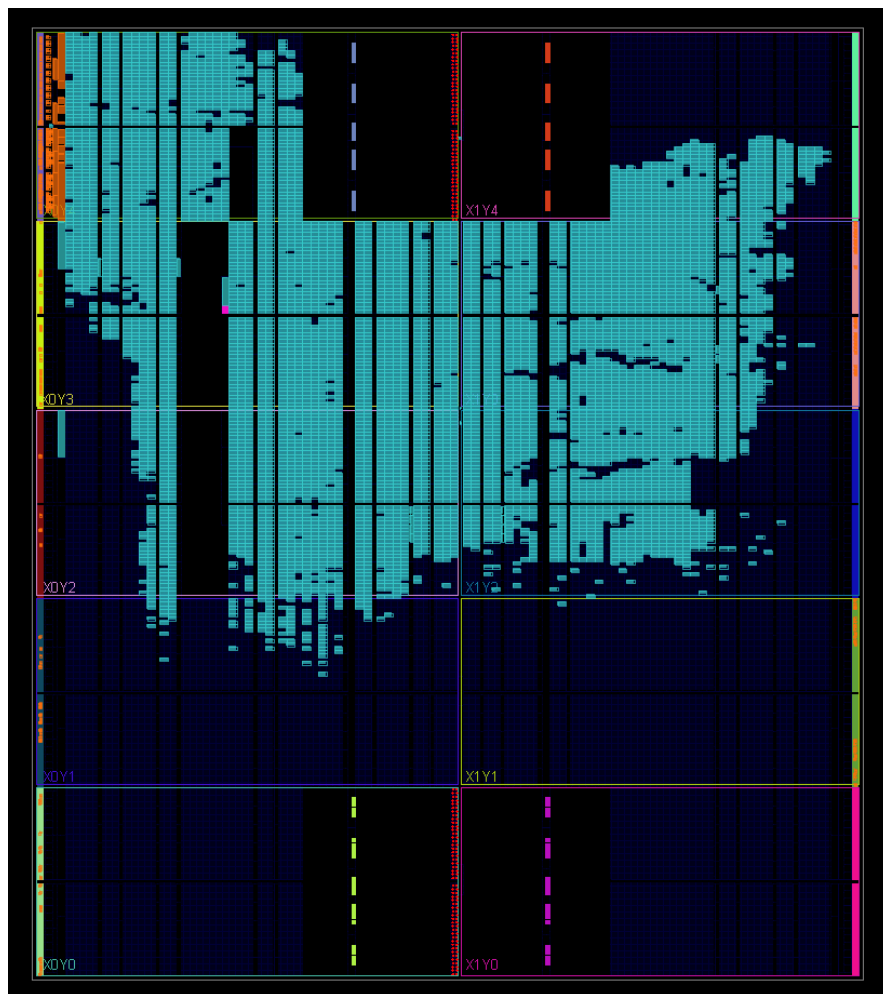
韩晓宇 charleshanxy@gmail.com

于凡奇 fan3002@gmail.com

力维辰 polar0228@gmail.com

马逸君 jasha@qq.com

项目简介：在该项目中，我们利用Vivado2019.2和SystemVerilog语言，设计了基于MIPS32指令集的处理器的，并在其上成功运行至Ucore系统的debug模式，及成功打印出Linux 版本信息。



目录

1	概述	3
1.1	项目背景	3
1.2	项目概览	3
1.3	实验环境	3
2	CPU结构	3
2.1	指令集	3
2.2	流水线结构	3
2.3	冲突处理	4
2.4	Cache	5
2.5	TLB	7
2.6	协处理器0	7
2.7	中断异常	7
3	系统软件	8
3.1	Pmon运行	8
3.2	Ucore内核启动	8
4	更多的工作	10
4.1	双发射	10
4.2	尝试启动Linux	10
5	声明与致谢	11
5.1	声明	11
5.2	致谢	11
6	参考书籍	11

1 概述

1.1 项目背景

本项目名为Vanilla CPU，是第四届龙芯杯决赛提交作品，由复旦大学计算机学院FDU1.2队四名成员耗时一个月共同完成。

1.2 项目概览

本项目主要分为两个部分，一个是CPU部分，一个是系统软件部分。

在CPU部分，我们设计了一个MIPS架构的五级流水线单发射CPU，同时支持Cache和TLB，能够通过大赛方给的全部测试，其中，决赛提交版本频率为95MHZ，IPC分数为28.690分。

在系统软件部分，我们将我们的CPU与官方给的文件相结合，成功搭建了一个soc，成功启动了Pmon，并运行至Ucore的debug界面及linux的版本信息部分。

1.3 实验环境

本实验基于Windows10.0.18362以及Ubuntu16.04系统下的vivado2019.2软件完成。

2 CPU结构

2.1 指令集

算数运算指令：ADD,ADDI,ADDU,ADDIU,SUB,SUBU,SLT,SLTI,SLTU,SLTIU,DIV,DIVU,MUL,MULT,MULTU,MADD,MADDU,MSUB,MSUBU,CLO,CLZ

逻辑运算指令：AND,ANDI,LUI,NOR,OR,ORI,XOR,XORI

移位指令：SLLV,SLL,SRAV,SRA,SRLV,SRL

分支跳转指令：BEQ,BNE,BGEZ,BGTZ,BLEZ,BLTZ,BGEZAL,BLTZAL,BEQL,BNEL,J,JAL,JR,JALR

数据移动指令：MFHI,MFLO,MTHI,MTLO,MOVN,MOVZ

访存指令：LB,LBU,LH,LHU,LW,SB,SH,SW,LWL,LWR,SWL,SWR

自陷指令：BREAK,SYSCALL

特权指令：ERET,MFC0,MTC0,TLBP,TLBR,TLBWI,TLBWR,WAIT

2.2 流水线结构

Vanilla CPU为MIPS架构的五级流水线单发射CPU，流水线可以分为取指，译码，执行访存，写回五个阶段。

在取指阶段，CPU从cache取出上一次发出的地址所对应的指令。

在译码阶段，流水线会对从取指阶段传下来的指令进行解析，一方面给出相应的控制信号，让流水线能够正确执行接下来几个阶段的各种操作，另一方面也通过当前指令及其pc值计算出下一条指令的地址，从而向cache发出正确的地址请求并在下一个周期取得正确的指令。

在执行阶段，CPU会执行运算相关的指令，并将执行的结果传给下一个阶段。对于部分指令，如果在计算的过程中触发了例外（如溢出等），则对应的例外信号也会传给下一个阶段进行处理。

在访存阶段，其主要工作有两个。一是执行访存操作，对内存（cache）进行读写。二是处理之前阶段产生的例外，给出相应的信号让流水线跳到异常处理处，并对cp0寄存器进行相应的操作。

在写回阶段，会将之前执行得到的数据写进通用寄存器和hi，lo寄存器内。

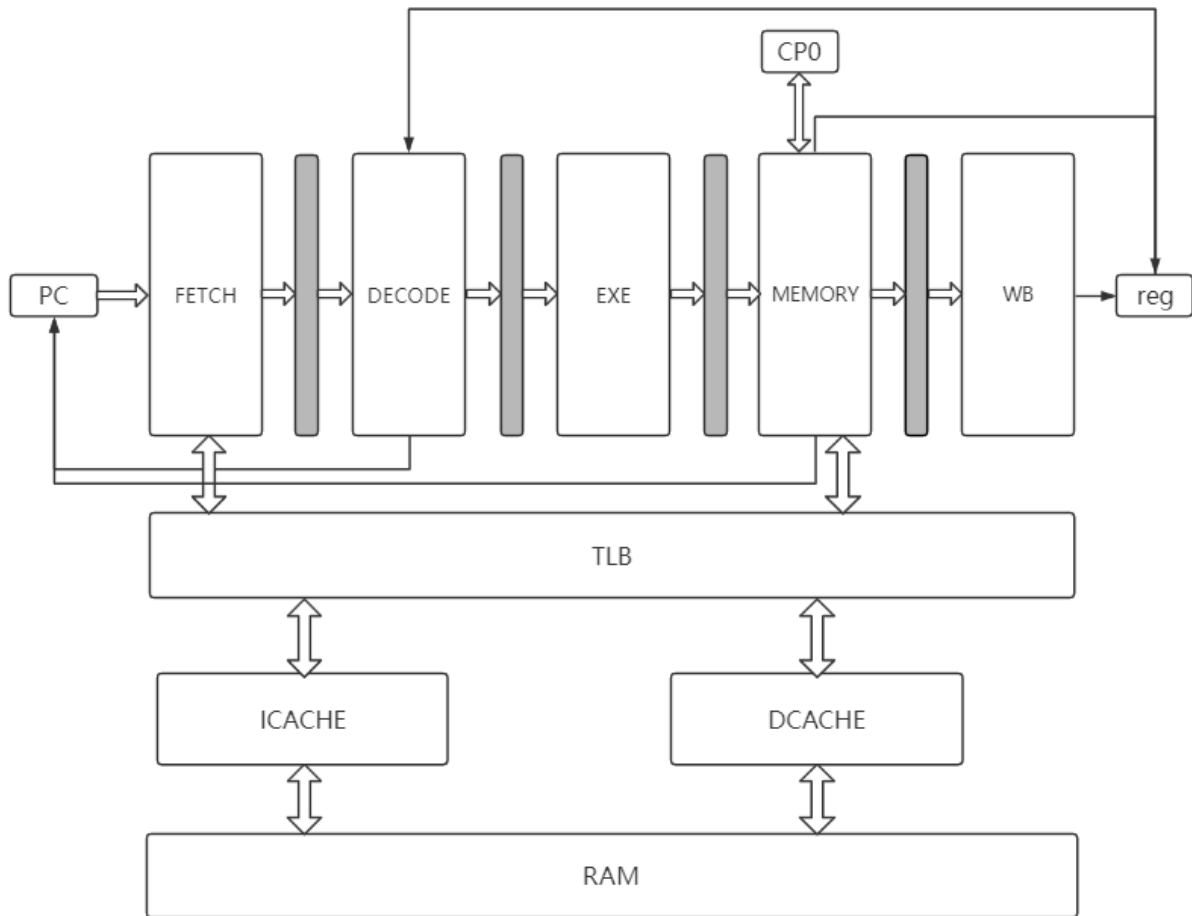


图 2-1 流水线结构

2.3 冲突处理

由于Vanilla CPU采用五级流水线结构，所以相邻的指令之间可能会产生数据冲突，即第一条指令需要写某一个寄存器，在还没有写进去的时候第二条指令便需要用到这个寄存器的值，这样就会导致第二条指令拿到错误的值进行计算，进而导致一连串的错误。因此，冲突处理模块接收各个阶段传来的信号，并向各个阶段发送避免冲突相关的信号，利用数据前递以及流水线暂停等方式来确保流水线指令运算的正确性。

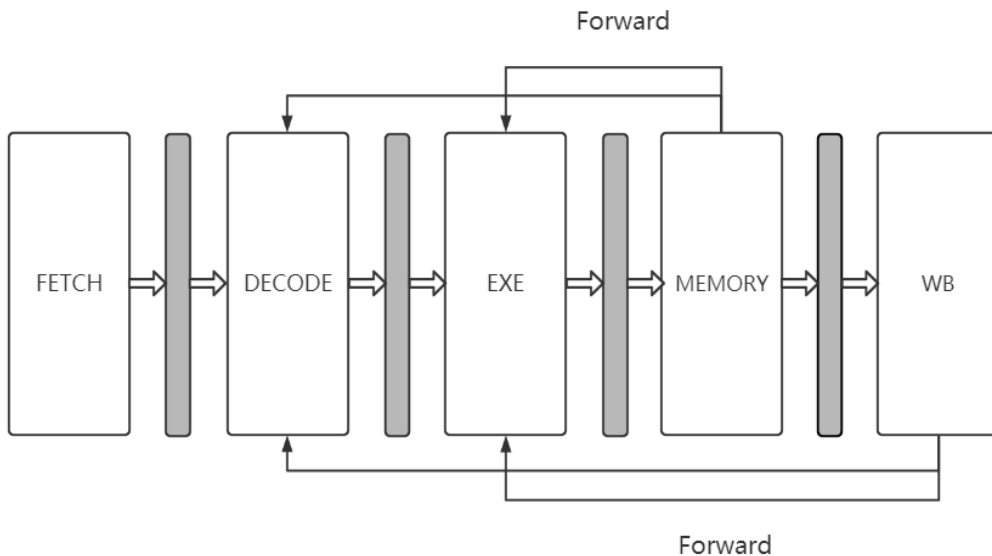


图 2-2 数据前递

2.4 Cache

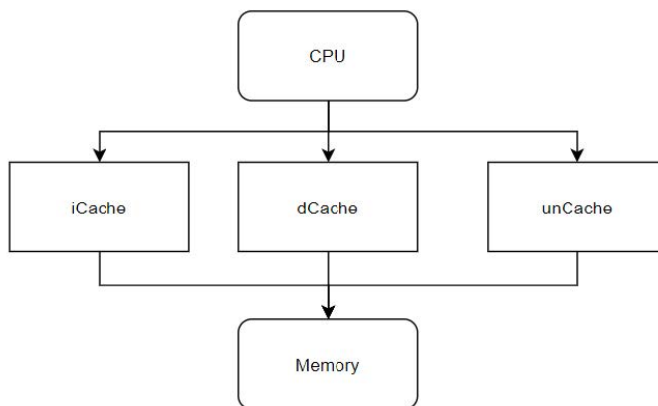


图 2-3 CPU与Memory的交互

CPU访存共两种模式，分为cached和uncached。其中，uncached模式下，CPU与memory 进行直接握手交互。在cached模式下，CPU和memory的交互通过cache实现，cache本身采用哈佛结构，将数据与指令分开进行缓存。在CPU和cache进行交互时，一旦cache命中，则cpu 可以在当周期获取到请求的数据，或者在当周期将数据写入可以在当周期获取到请求的数据，或者在当周期将数据写入cache。在和Memory的直接交互中，弃用了官方提供的的直接交互中，弃用了官方提供的AXI转接桥，在重构的转接桥中，利用组合逻辑实现用组合逻辑实现burst读写模式，大大提升了访存的效率。读写模式，大大提升了访存的效率。

在cache中，将数据内容和tag valid dirty等标签信息分开存储。当CPU请求cache时， cache根据地址 Index字段信息并行访问Data Ram和Info Ram，同时获取数据信息和标签信息，之后通过比较器判断是否命中，命中时在当周期将选择器选择出的数据返回给cpu。如果cache 未命中，则cache通过控制

器和替换策略进行数据读入、替换、写回等操作，暂停部分流水线，利用多个周期完成cache内部数据更新。

其中，icache为两周期，第一个周期判断是否命中，命中则直接返回，并且仍然维持第一个周期的状态，如果未命中则进入第二周期。第二个周期从内存中读取数据，icache向内存请求一次burst，通过多个时钟周期获得64字节的数据，然后返回第一周期。

dcache为三周期，第一周期判断是否命中，命中直接返回，并维持第一个周期的状态，未命中则根据替换策略判断是否需要写回。如果需要写回则进入第二周期，否则直接进入第三周期。在第二周期，dcache将数据写回，通过一次burst和多个时钟周期，将32字节的数据写回，然后进入第三周期。第三周期从内存中读取CPU请求的新数据，通过一次burst和多个周期，将32字节的数据读入，然后返回第一周期。

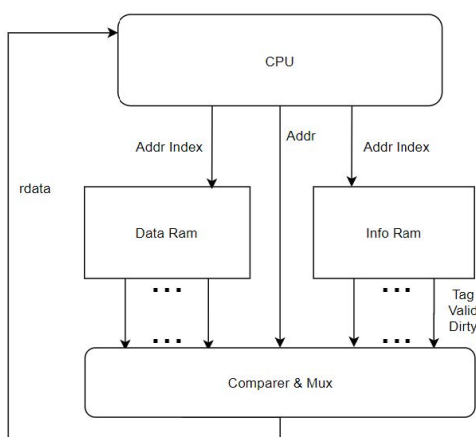


图 2-4 结构示意图

由于最简单的流水线cpu中，cpu和内存的交互占用了大量的时间，因此，在加上了 cache之后，除了未命中时仍需要和内存进行交互，其余时间可以当周期完成访存操作，从而性能测试有了显著的提升。在增加AXI burst支持后，由从前的每次访存都需要单独发请求变为连续地址访存可以通过一次请求完成，在原有基础上又有了相对显著的性能提升。

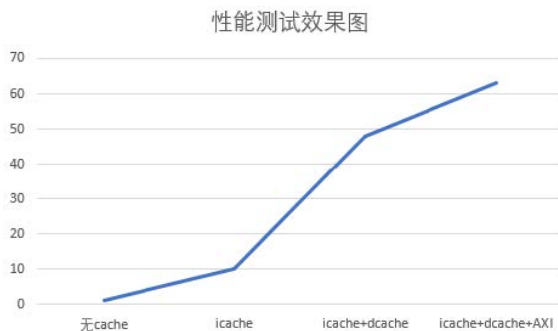


图 2-5 性能测试效果图

Number	Select	Name
0	0	Index
1	0	Random
2	0	EntryLo0
3	0	EntryLo1
4	0	Context
5	0	PageMask
6	0	Wired
8	0	BadVAddr
9	0	Count
10	0	EntryHi
11	0	Compare
12	0	Status
13	0	Cause
14	0	EPC
15	0	PRId
16	0	Config0
16	1	Config1

表 2-1 CP0寄存器

2.5 TLB

为了能够成功运行pmon和操作系统，我们也实现了一个标准的大小为4KB，32项的TLB模块，用于帮助我们将虚拟地址转换成物理地址。同时，我们实现了4条TLB指令和必要的TLB例外使得软件能够直接对我们的 TLB进行初始化等操作。

2.6 协处理器0

CP0，即协处理器0，在CPU设计中扮演着重要的功能。其可以协助配置CPU工作状态，进行高速缓存控制，异常控制，存储管理单元控制等。在表2-1中，我们列出我们在CPU设计中所用到的CP0寄存器。

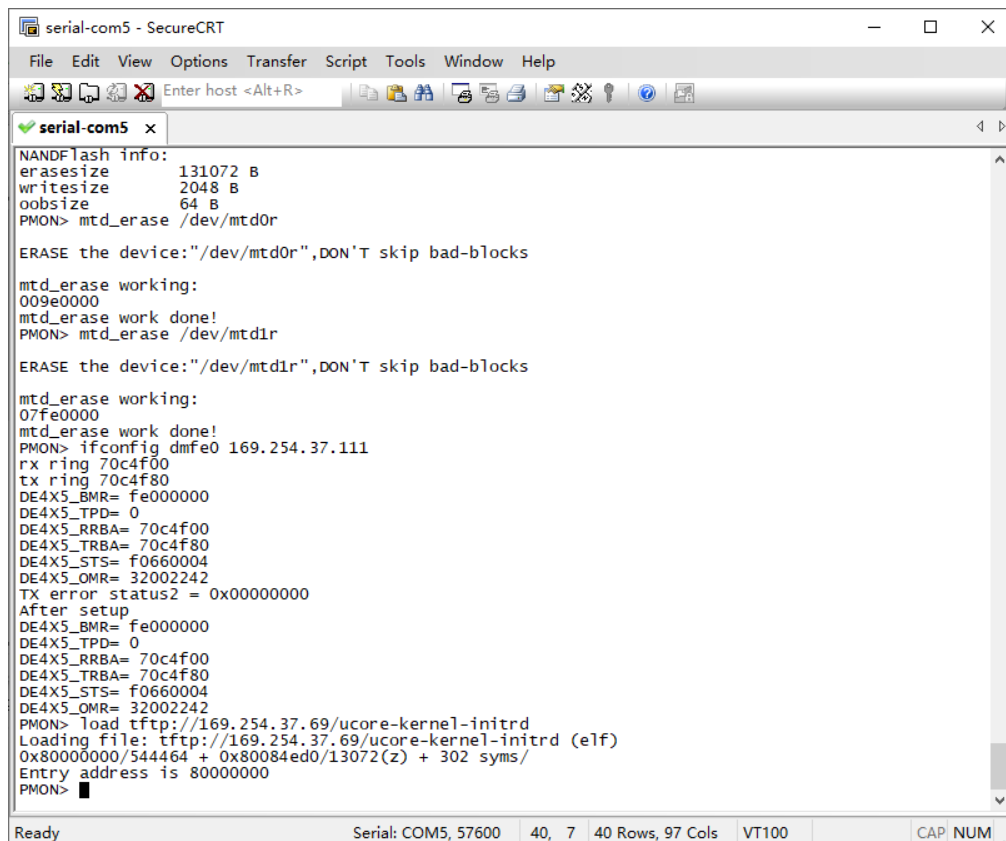
2.7 中断异常

按照本次大赛及启动系统的部分要求，我们实现了中断，取指产生的地址错例外，保留指令例外，整型溢出例外、陷阱例外、系统调用例外，数据访问产生的地址错例外，TLB Refill 例外，TLB Invalid例外和TLB Modified例外。在触发这些中断例外的时候，会先判断优先级，然后流水线会跳转到对应的例外处理地址去处理这些中断例外，处理完之后通过eret跳转回触发中断例外地址的下一条地址。

3 系统软件

3.1 Pmon运行

在运行系统之前，我们首先需要运行pmon作为一个小的bios程序。在此处我们使用了自己剪裁的pmon，用其生成了与大赛方不同的gzrom.bin，并将其通过串口软件传输到实验板上，传输完成后，烧写通过官方提供的soc平台和我们自己的cpu搭建生成出来的 bitstream文件，复位实验板后加载一段时间，便能成功进入pmon。我们同时尝试了一些pmon 中的命令，如下图，均能正常运行，因此我们认为到此为止，pmon已经处于正常运行状态。



```
serial-com5 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
serial-com5 x
NANDFlash info:
erasesize 131072 B
writesize 2048 B
oobsize 64 B
PMON> mtd_erase /dev/mtd0r
ERASE the device:"/dev/mtd0r",DON'T skip bad-blocks
mtd_erase working:
009e0000
mtd_erase work done!
PMON> mtd_erase /dev/mtd1r
ERASE the device:"/dev/mtd1r",DON'T skip bad-blocks
mtd_erase working:
07fe0000
mtd_erase work done!
PMON> ifconfig dmfe0 169.254.37.111
rx ring 70c4f00
tx ring 70c4f80
DE4X5_BMR= fe000000
DE4X5_TPD= 0
DE4X5_RRBA= 70c4f00
DE4X5_TRBA= 70c4f80
DE4X5_STS= f0660004
DE4X5_OMR= 32002242
TX error status2 = 0x00000000
After setup
DE4X5_BMR= fe000000
DE4X5_TPD= 0
DE4X5_RRBA= 70c4f00
DE4X5_TRBA= 70c4f80
DE4X5_STS= f0660004
DE4X5_OMR= 32002242
PMON> load tftp://169.254.37.69/ucore-kernel-initrd
Loading file: tftp://169.254.37.69/ucore-kernel-initrd (elf)
0x80000000/544464 + 0x80084ed0/13072(z) + 302 syms/
Entry address is 80000000
PMON> █
Ready Serial: COM5, 57600 40, 7 40 Rows, 97 Cols VT100 CAP NUM
```

图 3-6 Pmon运行效果图

3.2 Ucore内核启动

在启动了PMON的基础上，我们将自己生成的ucore-kernel-initrd通过tftp传至开发板上并启动。此后程序能够打印出寄存器状态信息及系统的初始化情况，最终进入内核debug状态。在此状态下我们可以使用help指令显示可用的指令(kerninfo)，并能用kerninfo打印出内核信息。


```

serial-com5 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
serial-com5 x
DE4X5_TPD= 0
DE4X5_RRBA= 70c4f00
DE4X5_TRBA= 70c4f80
DE4X5_STS= f0660004
DE4X5_OMR= 32002242
PMON> load tftp://169.254.37.69/ucore-kernel-initrd-2
Loading file: tftp://169.254.37.69/ucore-kernel-initrd-2 (e1f)
0x80000000/556192 + 0x80087ca0/13072(z) + 302 syms\
Entry address is 80000000
PMON> g console=ttys0, 57600 rdinit=sbin/init
zero at v0 a0 a1 a2 a3
00000000 00000000 00000000 00000000 00000004 a7dff78 a7dff78c 870bacc0
t0 t1 t2 t3 t4 t5 t6 t7
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
s0 s1 s2 s3 s4 s5 s6 s7
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
t8 t9 k0 k1 gp sp s8 ra
00000000 00000000 00000000 00000000 00000000 a7dff78 00000000 8707a668
tlb invalidated
console is inited
++setup timer interrupts
Initrd: 0x8002c6d0 - 0x80083ecf, size: 0x00057800, magic: 0x2f8dbe2a
(THU.CST) os is loading ...

Special kernel symbols:
entry 0x80000104 (phys)
etext 0x8002A400 (phys)
edata 0x80087CA0 (phys)
end 0x8008AFB0 (phys)
kernel executable memory footprint: 387KB
memory management: buddy_pmm_manager
memory map:
[80000000, 82000000]

freemem start at: 800C8000
free pages: 00001F35
## 00000020
check_alloc_page() succeeded!
check_pgdir() succeeded!
kernel panic at kern/mm/pmm.c:412:
assertion failed: *(int*)0x100 == 0x1234
welcome to the kernel debug monitor!!
Type 'help' for a list of commands.
help - Display this list of commands.
kerninfo - Display information about the kernel.
Special kernel symbols:
entry 0x80000104 (phys)
etext 0x8002A400 (phys)
edata 0x80087CA0 (phys)
end 0x8008AFB0 (phys)
kernel executable memory footprint: 387KB
k>
Ready Serial: COM5, 57600 52, 4 52 Rows, 97 Cols VT100 CAP NUM

```

图 3-7 Ucore运行效果图

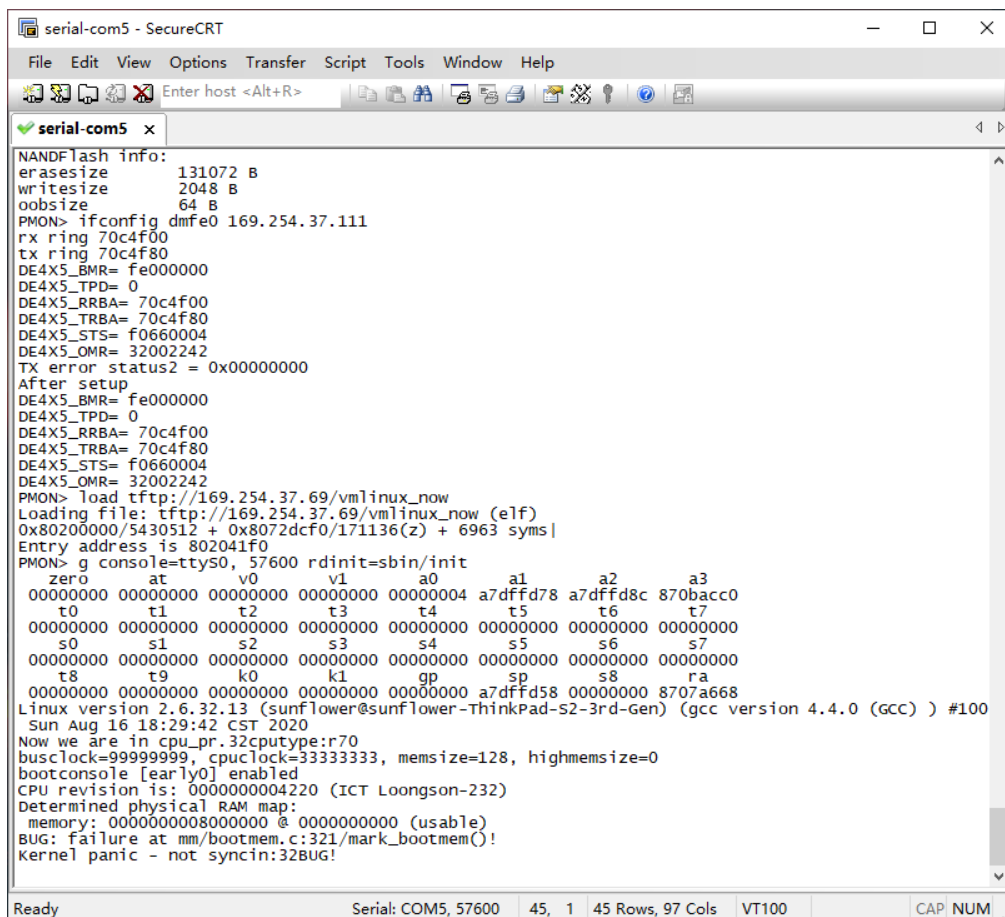
4 更多的工作

4.1 双发射

在初赛中，我们尝试并成功实现了双发射，经过验证可以通过初赛所有的测试，并能在性能测试中取得约40分的成绩，但由于发射阶段的逻辑比较繁杂，导致频率相较于经优化的单发射而言较低，且因为没有过多的时间分更多的流水来提升频率，因此我们的最终版本并没有采用双发射的版本，而是采取了得分更高并且更好调试的五级单发射版本。倘若以后有时间则会尝试进一步完善双发射版本并冲击更高的分数。

4.2 尝试启动Linux

由于我们成功地运行了pmon，我们又尝试启动自行编译的Linux内核（删除了部分未实现的指令），不过在运行的时候卡在了下图处未能成功运行，鉴于没有充足的时间，最后也没有能够成功解决问题，故并没有成功启动Linux，仅做了一些有意义的尝试。



```

serial-com5 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
serial-com5 x
NANDFlash info:
erasesize      131072 B
writesize      2048 B
oobsize        64 B
PMON> ifconfig dmfe0 169.254.37.111
rx ring 70c4f00
tx ring 70c4f80
DE4X5_BMR= fe000000
DE4X5_TPD= 0
DE4X5_RRBA= 70c4f00
DE4X5_TRBA= 70c4f80
DE4X5_STs= f0600004
DE4X5_OMR= 32002242
TX error status2 = 0x00000000
After setup
DE4X5_BMR= fe000000
DE4X5_TPD= 0
DE4X5_RRBA= 70c4f00
DE4X5_TRBA= 70c4f80
DE4X5_STs= f0600004
DE4X5_OMR= 32002242
PMON> load tftp://169.254.37.69/vmlinux_now
Loading file: tftp://169.254.37.69/vmlinux_now (elf)
0x80200000/5430512 + 0x8072dcf0/171136(z) + 6963 syms|
Entry address is 802041f0
PMON> g console=ttys0, 57600 rdinit=sbin/init
zero at v0 v1 a0 a1 a2 a3
00000000 00000000 00000000 00000000 00000004 a7dff78 a7dff8c 870bacc0
t0 t1 t2 t3 t4 t5 t6 t7
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
s0 s1 s2 s3 s4 s5 s6 s7
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
t8 t9 k0 k1 gp sp s8 ra
00000000 00000000 00000000 00000000 00000000 a7dff58 00000000 8707a668
Linux version 2.6.32.13 (sunflower@sunflower-ThinkPad-S2-3rd-Gen) (gcc version 4.4.0 (gcc) ) #100
Sun Aug 16 18:29:42 CST 2020
Now we are in cpu_pr.32cputype:r70
busclock=99999999, cpuclock=33333333, memsize=128, highmemsize=0
bootconsole [early0] enabled
CPU revision is: 000000004220 (ICT Loongson-232)
Determined physical RAM map:
 memory: 000000008000000 @ 0000000000 (usable)
BUG: failure at mm/bootmem.c:321/mark_bootmem()!
Kernel panic - not syncin:32BUG!
Ready Serial: COM5, 57600 45, 1 45 Rows, 97 Cols VT100 CAP NUM

```

图 4-8 Linux运行效果图

5 声明与致谢

5.1 声明

本组有关本次大赛的所有代码，文档均放在github上做开源处理，如需使用注明即可。

5.2 致谢

感谢本组四位同学一个月以来的奋斗与努力，感谢FDU1.1队提供的帮助，感谢复旦大学计算机学院的大力支持，感谢张亮老师和陈辰老师的帮助与鼓励，同时感谢北邮、清华、国科大、北理工的开源代码以及文档为我们几位新手提供的指引与帮助，最后感谢龙芯官方为举办本次比赛所做的一切工作！

6 参考书籍

Digital Design and Computer Architecture - David Money Harris & Sarah L. Harris

See MIPS Run - Dominic Sweetman

兼容ARM9的软核处理器设计基于FPGA - 李新兵

自己动手写CPU - 雷思磊