B62007Y 2017-2018学年春季学期

计算机组成原理实验

实验项目5 联合大实验

“三个实验内容至少完成一个”

主讲教师： 张 科

2018年6月8日

中国科学院大学 University of Chinese Academy of Sciences
中科院计算所 INSTITUTE OF COMPUTING TECHNOLOGY,CAS

实验内容

- 基于实验项目4实现的处理器，至少选择并完成下列实验内容之一（可多选，但不能不选）

编号	实验项目名称	实验内容简介
5.1	复杂处理器设计	完成多周期处理器、Cache和流水线处理器硬件设计， 并进行性能评估对比
5.2	DMA引擎设计	实现基于队列结构的DMA硬件硬件逻辑； 在处理器中添加中断机制并实现DMA中断处理； 完成软-硬件协同的块数据DMA搬移及性能评估
5.3	深度学习算法 及硬件加速	完成卷积算法的C语言软件实现； 在处理器中添加乘法指令，提升算法处理性能； 添加卷积算法硬件加速器及控制软件

实验项目进度安排

❑ 课堂验收

- 截止日：2018年07月06日20:49:59
- 检查同学们选择完成实验项目的上板运行结果及代码实现

❑ 最终提交

- 截止日：2018年07月13日18:09:59
- 提交内容包括所选实验项目的全部RTL源码、C软件程序源码、运行日志文件及实验报告

B62007Y 2017-2018学年春季学期

计算机组成原理实验

实验项目5.1 复杂处理器设计

常轶松 黄博文

2018年6月8日

中国科学院大学
University of Chinese Academy of Sciences

中科院计算所
INSTITUTE OF COMPUTING TECHNOLOGY,CAS

内容大纲

- 实验内容简介及要求
- 实验要点讲解
- 实验操作流程
- 注意事项

实验目的与要求

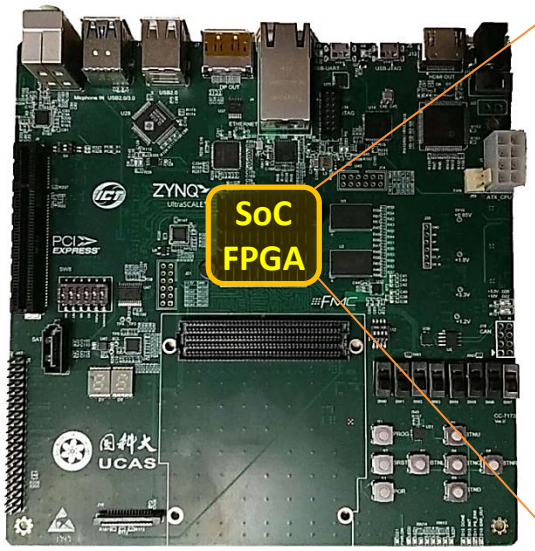
❑ 基于实验项目4实现的MIPS处理器

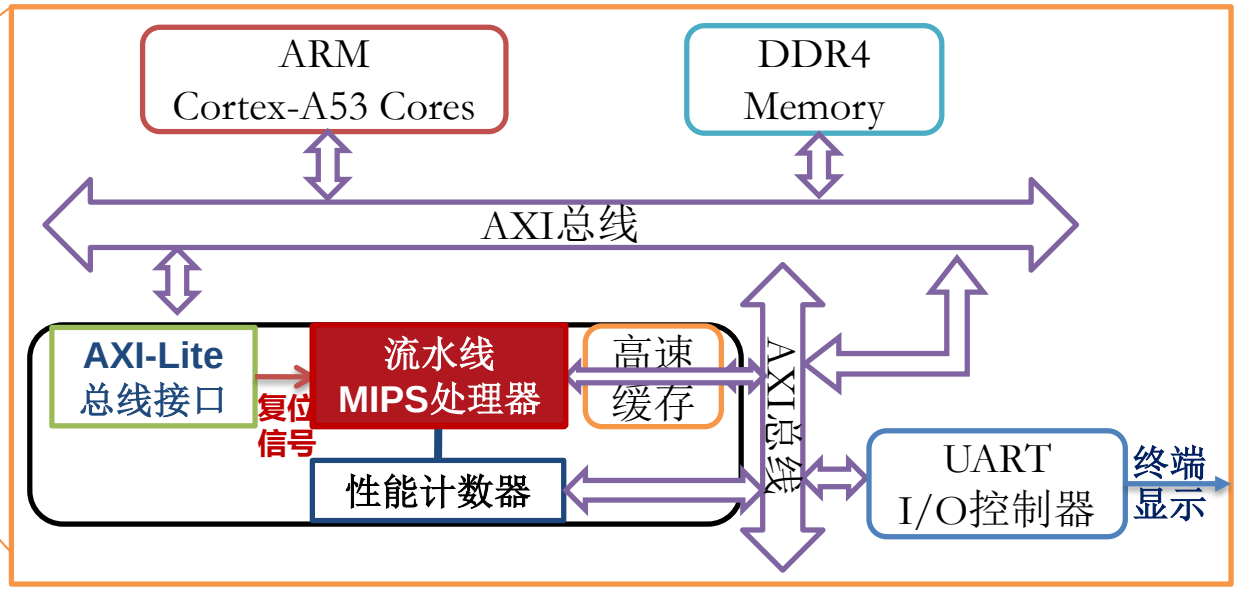
- 修改状态机，实现完整多周期处理器
- 添加指令Cache和数据Cache，解决处理器访存性能瓶颈
- 初步尝试实现基本流水线结构
- 充分利用性能计数器，评估不同设计的性能指标
- 优化逻辑实现，提高处理器主频

❑ 实验要求

- 至少完成多周期处理器设计
- 可选做Cache和流水线结构

实验环境及工程框架





- 设计完整多周期处理器
  - 使用性能计数器对medium和advanced两组测试用例进行性能评估
- 为处理器添加Cache，完成Cache核心替换算法的硬件实现
  - 统计访存延时变化
- 实现流水线结构处理器
  - 统计性能变化情况，并尝试进一步优化流水结构

实验项目接收

- 实验项目5.1参与邀请链接URL（预计本周末发布）
  - **链接地址以SEP网站上发布的最终版ppt显示的为准**
- 克隆个人本地仓库
  - git clone https://github.com/ucas-cod-18sp/prj5-cpu-opt-<USERNAME>
- 在个人本地仓库中添加本次实验的原始框架仓库地址
  - **git remote add upstream https://github.com/ucas-cod/prj5-cpu-opt-student**
  - 更新同步方法请参考之前实验项目讲义

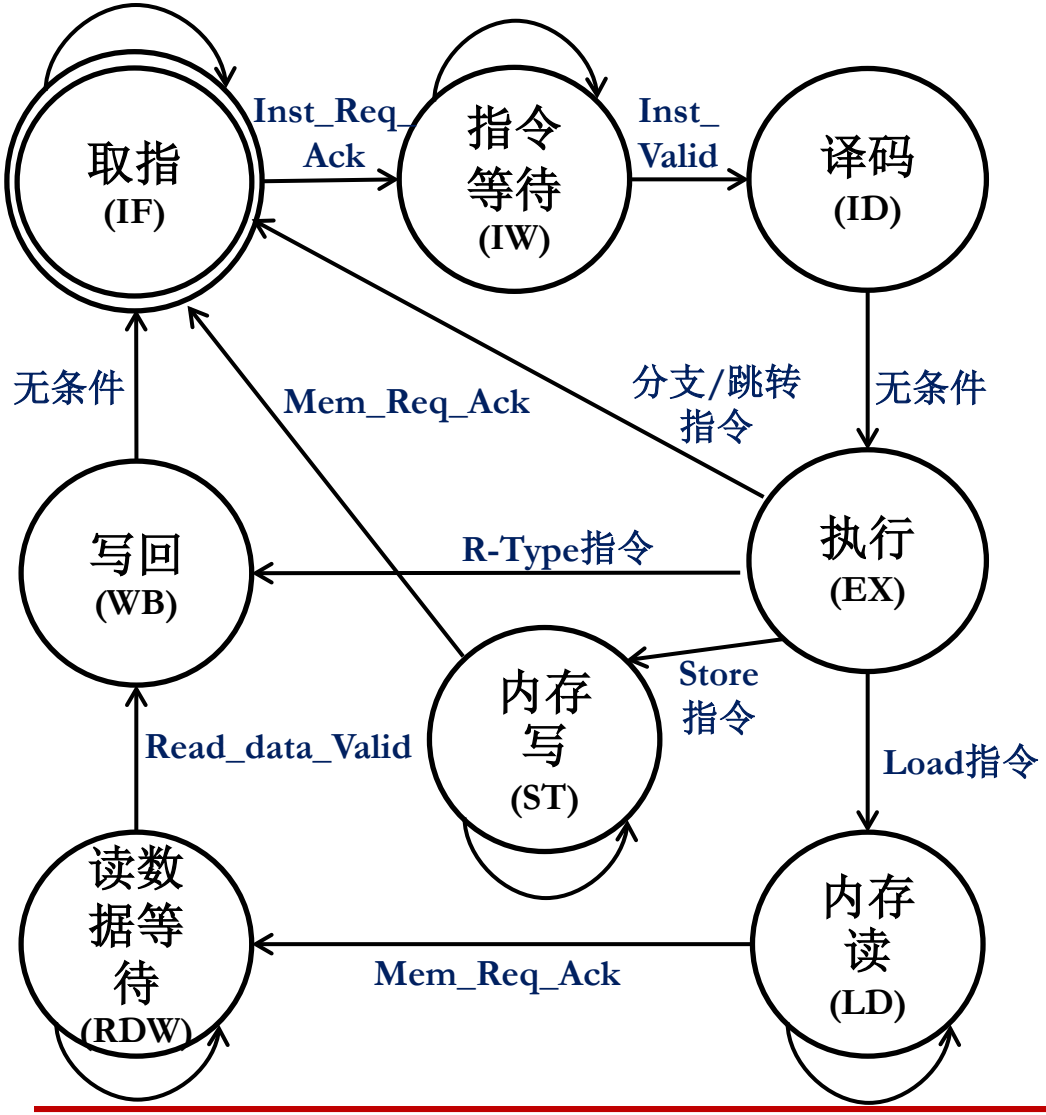
实验流程要求

- **阶段1：在个人本地仓库中的master分支中完成多周期处理器实验**
- **阶段2：在完成多周期处理器设计实验后，开始Cache设计实验**
  - git checkout –b cache创建并切换到本阶段实验的工作分支
  - git pull upstream cache同步该阶段的脚本及代码框架
- **阶段3：在完成Cache设计实验后，再开始流水线设计**
  - git checkout –b pipeline创建并切换到本阶段实验的工作分支

内容大纲

- 实验内容简介及要求
- 实验要点讲解
  - 复杂处理器设计
  - Benchmark修改
- 实验操作流程
- 注意事项

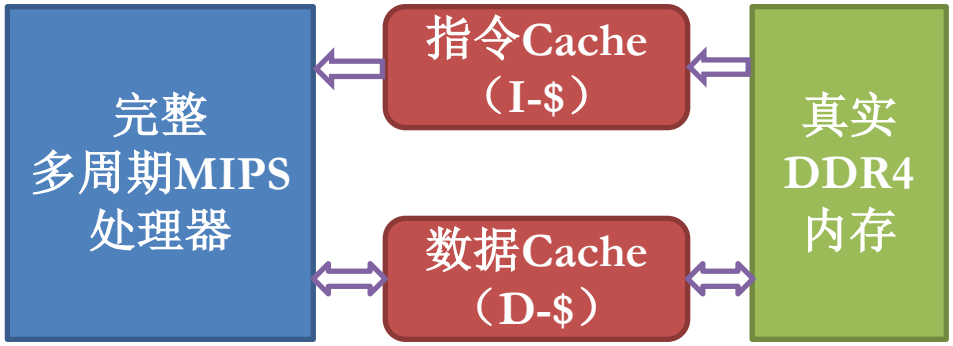
完整的多周期处理器状态机



- 将译码（ID）和执行（EX）拆分成两个处理状态
- 分支/跳转指令在执行阶段（更新PC值）后立即进入下一条指令的取指阶段
  - JAL指令要在EX阶段将返回地址写入r31寄存器
- R-Type类型指令在执行阶段后进入写回（WB）阶段
  - R-Type跳转指令（JR、JALR）在EX阶段完成PC值更新
  - JR指令无需写回操作，在WB阶段空等一拍
  - JALR指令在WB阶段将返回地址写入r31寄存器
- 其他状态与实验项目4中实现的多周期处理器状态机要求保持一致

Cache基本结构

- **为处理器添加独立的指令和数据Cache**
  - 这种结构称为Harvard结构
  - 为后续流水线实现消除结构相关



Cache结构参数	I-$	D-$
Cache容量（Byte）	512	
组织方式	4-路组相连	
Cache Line大小（Byte）	16	
写命中策略	--	Write-back
写缺失策略	--	Write Non-allocate

Cache和处理器及内存接口的数据位宽为32-bit（4 Byte）

- **实验要求**
  - 添加Cache核心替换算法的硬件实现
- **Cache的详细结构及代码修改方案安排在下周上课时间（6月15日）介绍**
  - Cache代码也会在下周发布

面向真实内存的流水线处理器设计

- **设计7级流水线结构，实现单发射顺序执行处理器**
  - **IF**：访问指令Cache，等待（阻塞等待，直到指令Cache接收请求，暂停流水线，等待结束可更新PC）
  - **IW**：等待指令Cache返回指令码（阻塞等待，暂停流水线）
  - **ID**：指令译码，同时完成跳转目标地址及跳转条件计算，并将计算结果前馈到IF；如译码出指令是跳转指令且跳转发生，清空IF和IW流水级处理的指令
  - **EX**：访存地址计算及ALU运算
  - **MEM**：访存类指令访问数据Cache（阻塞等待，直到数据Cache接收请求，暂停流水线）；其他指令空等一拍
  - **RDW**：Load指令等待数据Cache返回读数据（阻塞等待，暂停流水线）；其他指令空等一拍
  - **WB**：寄存器结果写回

内容大纲

- 实验内容简介及要求
- 实验要点讲解
  - 复杂处理器设计
  - Benchmark修改
- 实验操作流程
- 注意事项

实验项目使用的Benchmark

- **实验项目5.1使用medium和advanced两组benchmark**
  - 编写UART访问驱动
  - 编写性能计数器访问驱动
  - 修改benchmark各测试程序的main()函数，添加计数器访问和结果打印功能

- **通过在main()函数的“return 0”之前打印性能统计结果，除完成性能评估外，还可用于判断benchmark程序是否正确执行**
  - 当添加Cache后，ARM将无法判断MIPS是否正确执行了benchmark，此时可通过MIPS的输出结果直接判断

库函数修改

- **UART访问驱动**
  - 将实验项目4中使用的printf.c复制到个人仓库的benchmark/lib/src下
  - UART控制器基地址为0x40010000

- **性能计数器访问驱动（与实验项目4中的修改类似）**
  - 修改benchmark/lib/include/perf_cnt.h的Result结构体，添加统计量
  - 修改benchmark/lib/src/perf_cnt.c的bench_prepare()和bench_done()函数，通过调用_uptime()将性能结果存入Result结构体统计变量中
  - 修改benchmark/lib/src/perf_cnt.c的_uptime()函数，访问性能计数器
  - 性能计数器的物理地址与实验项目4相同

Benchmark修改

- 请修改benchmark/advanced/src/*.c及benchmark/medium/src/*.c，添加性能计数和结果打印功能
  - 右图以benchmark/advanced/src/load-store.c源码为例，对main()函数修改方法进行说明

思考：如果想在main()函数中使用当前C源码文件中未定义声明的bench_prepare()、bench_done()和printf()，还要做什么？

```
int main() {
  unsigned i;

  Result res;
  res.msec = 0;

  bench_prepare(&res);

  for(i = 0; i < ARR_SIZE(mem) - 2; i ++) {
    nemu_assert((short)mem[i] == lh_ans[i]);
  }

  for(i = 0; i < ARR_SIZE(mem) - 2; i ++) {
    nemu_assert(mem[i] == lhu_ans[i]);
  }

  for(i = 0; i < ((ARR_SIZE(mem) - 2) / 2); i ++) {
    unsigned x = ((unsigned*)((void*)mem + 1))[i];
    nemu_assert(x == lwlr_ans[i]);
  }

  for(i = 0; i < ARR_SIZE(mem) - 2; i ++) {
    mem[i] = ~(1 << (2 * i + 1));
    nemu_assert(mem[i] == sh_ans[i]);
  }

  bench_done(&res);

  printf("total cycle %d\n", res.msec);

  return 0;
}
```

声明Result结构体对象，读计数器初值

获取性能结果并打印

内容大纲

- 实验内容简介及要求
- 实验要点讲解
- 实验操作流程
  - 硬件实验流程
  - 软件编译流程
  - 上板运行流程
- 注意事项

处理器RTL代码编写

- **请在本地仓库hardware/sources/ip_catalog/mips_core目录下实现处理器代码**
  - 将实验项目4中对应目录下的代码全部拷贝过来
  - 添加完整多周期处理器实现代码

硬件操作命令

- **本次实验项目使用的Vivado硬件流程命令与之前实验项目基本一致，在此不再赘述。请参考本次实验项目仓库中的README**
  - 注意：DDR内存无法在仿真环境提供，所以功能仿真和时序仿真只能使用BRAM作为MIPS访问的真实内存

- **bit_gen命令可设置处理器工作频率（默认工作频率100MHz）**
  - 例如：可执行make HW_ACT=bit_gen HW_VAL=120 vivado_prj设置处理器工作频率为120MHz
  - 如需在100MHz默认工作频率下添加硬件探针，需要在探针模块名列表前加一个冒号，例如：
    make HW_ACT=bit_gen HW_VAL="":mips_cpu_dbg" vivado_prj
  - 如在其他频率下进行硬件debug，需要在冒号前加上期望工作频率，例如：
    make HW_ACT=bit_gen HW_VAL="120:mips_cpu_dbg" vivado_prj即可在120MHz处理器工作频率下添加硬件探针
  - 时序仿真需要的时钟频率请自行修改testbench和hardware/constraints/mips_cpu_simu.xdc约束文件中的时钟定义

内容大纲

- 实验内容简介及要求
- 实验要点讲解
- 实验操作流程
  - 硬件实验流程
  - 软件编译流程
  - 上板运行流程
- 注意事项

MIPS测试软件程序编译

❑ 软件程序交叉编译

- 在个人本地仓库中分别执行
  make medium_bench和make advacned_bench对两组benchmark进行编译，生成的可执行文件分别位于benchmark/medium/bin目录和benchmark/advanced/bin下
- 如编译后再次修改C语言源码文件，需重新执行上述命令，重新生成可执行程序
- 执行make medium_bench_clean或make advanced_bench_clean，将删除编译生成的对应benchmark组下的二进制可执行文件

内容大纲

- 实验内容简介及要求
- 实验要点讲解
- 实验操作流程
  - 硬件实验流程
  - 软件编译流程
  - 上板运行流程
- 注意事项

本地与云端上板测试运行

- **在虚拟机的本地仓库中执行**

  make BOARD_IP=<ip_address> HW_VAL="<benchmark列表>" local_run

  连接ZyForce本地板卡或

  make USER=<user_name> HW_VAL="<benchmark列表>" cloud_run

  连接ZyForce云端远程板卡

  - 注意：上述命令中的引号不能去掉
  - <benchmark列表> = <benchmark组名>[:<benchmark组内序列号列表>]
    - []内的内容是可选项，不是必须填写。[]在实际命令中不出现
  - <benchmark组名>：medium/advanced二选一
  - 如果<benchmark组内序列号列表>为空，表示要运行某一组内的全部测试程序
    - 例如：make BOARD_IP=192.168.100.20 HW_VAL="medium" local_run
  - 如果<benchmark组内序列号列表>不为空，请在冒号后添加要运行的测试程序序列号（可按任意顺序且可重复），序列号之间用空格分隔
    - 例如：make USER=zhangsan HW_VAL="advanced:01 09 05 12" cloud_run
    - 组内序列号有效值：medium组01-12；advanced组01-17
    - 注意：不要忘记组名和序号列表之前的冒号（英文符号）
    - 注意：个位数的序列号前请务必加0（如序列号5务必输入为05）

内容大纲

- 实验内容简介及要求
- 实验要点讲解
- 实验操作流程
- 注意事项

实验项目5.1完整提交内容

- 各阶段代码及运行结果远程提交
  - 完整多周期处理器：git push origin master
  - Cache设计：git push origin cache
  - 流水线处理器：git push origin pipeline
- 实验报告
  - 请在实验项目5.1个人本地仓库的master分支中创建顶层目录doc（仅需执行一次）
    - mkdir –p doc
  - 将实验报告的PDF版本放入doc目录，命名规则为“学号-prj5-1.pdf”，例如：
    - 2015K8009929000-prj5-1.pdf
  - 执行git add –all && git commit –m “doc: project report”完成本地提交