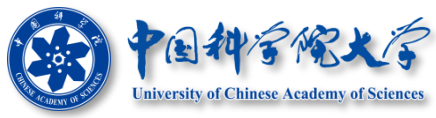
❑请在“国科大在线”APP签到

B0911007Y-01/02　2021-2022学年春季学期

计算机组成原理（研讨课）

实验项目4　定制RISC-V功能型处理器设计

2022年4月29日



0.1 实验内容与实验目的

- **基于实验项目3已实现的微结构，设计支持RISC-V 32-bit 整型指令集（RV32I）的功能型处理器**
  - 通过阅读手册，初步了解新兴RISC-V指令集
  - 进一步理解理论课讲授的精简指令集架构
  - **通过功能和性能评估，**对比RISC-V/MIPS指令集译码器的实现开销，理解RISC-V指令格式的设计思想

0.2 实验项目进度安排

课程内容	2022 4.15	4.22	4.29	5.6	5.13	5.20	5.27	5.30 (周一)
实验项目内容	实验发布		课堂讲解					
课堂验收							截止时间 18:59:59	
最终推送								截止时间 23:59:59

- 本次实验不设阶段提交
- 自5月6日上课时间起开始进行课堂验收
- 课堂验收截止：5月27日下课时（18:59:59）
- 最终提交截止时间前（5月30日周一23:59:59）需提交完整Verilog HDL代码及实验报告

RISC-V 32-bit整型指令集（RV32I）

The RISC-V Instruction Set Manual
Volume I: User-Level ISA
Document Version 2.2

Editors: Andrew Waterman[1], Krste Asanović[1,2]
[1]SiFive Inc.,
[2]CS Division, EECS Department, University of California, Berkeley
andrew@sifive.com, krste@berkeley.edu
May 7, 2017

RISC-V指令集手册（已上传至SEP网站）

31 27	26 25	24 20	19 15	14 12	11 7	6 0	
funct7		rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]			rs1	funct3	rd	opcode	I-type
imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]					rd	opcode	U-type
imm[20|10:1|11|19:12]					rd	opcode	J-type

与MIPS相比，RISC-V具有更加规整的指令格式

指令格式及指令集详见指令手册第104页

具体指令语义请查看指令手册第2章

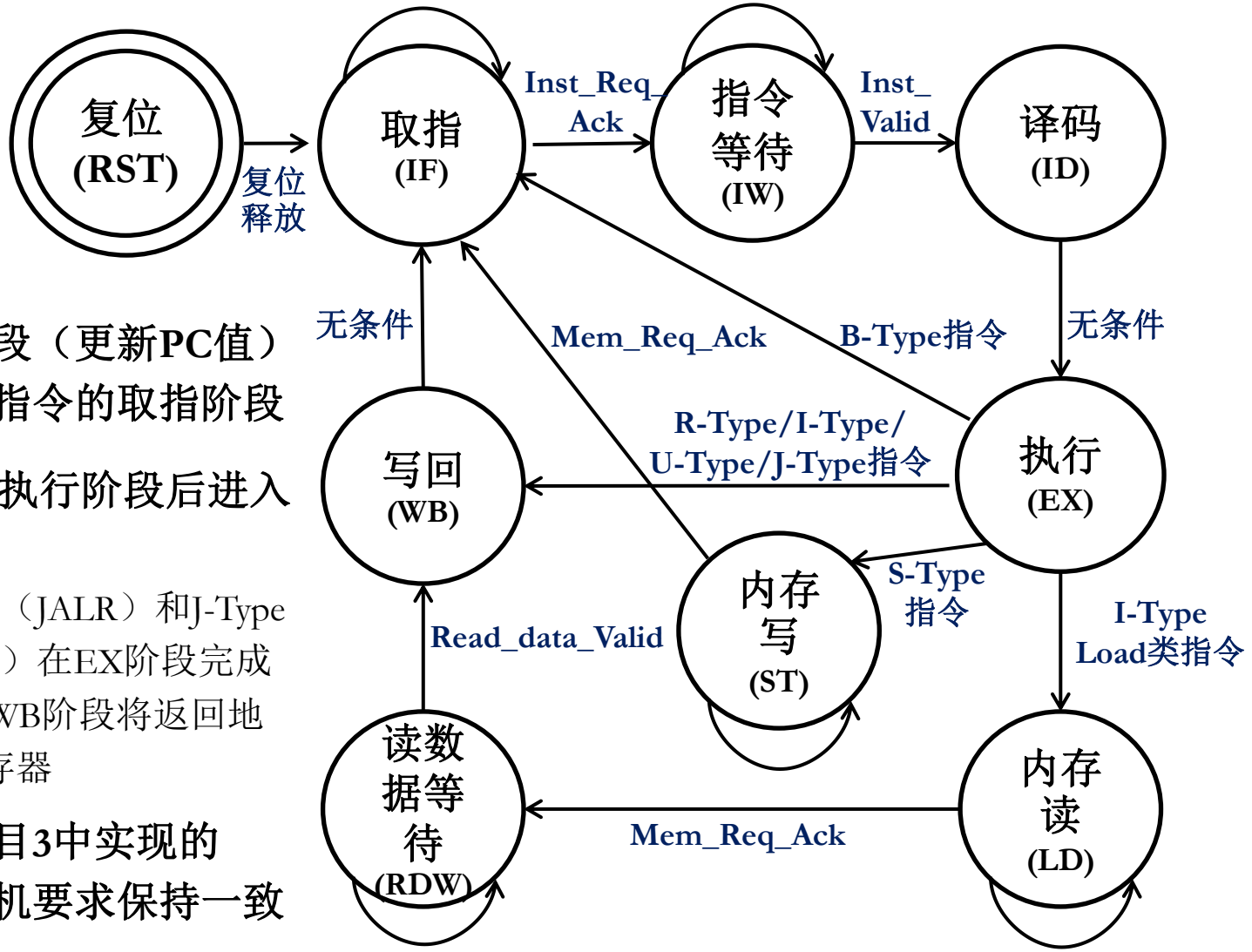
本次实验要实现的37条基本指令

RV32I Base Instruction Set

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND
0000	pred succ	00000	000	00000	0001111	FENCE
0000	0000 0000	00000	001	00000	0001111	FENCE.I
000000000000		00000	000	00000	1110011	ECALL
000000000001		00000	000	00000	1110011	EBREAK
csr		rs1	001	rd	1110011	CSRRW
csr		rs1	010	rd	1110011	CSRRS
csr		rs1	011	rd	1110011	CSRRC
csr		zimm	101	rd	1110011	CSRRWI
csr		zimm	110	rd	1110011	CSRRSI
csr		zimm	111	rd	1110011	CSRRCI

处理器状态机

- 分支指令在执行阶段（更新PC值）后立即进入下一条指令的取指阶段
- R-Type类型指令在执行阶段后进入写回（WB）阶段
  - I-Type跳转指令（JALR）和J-Type跳转指令（JAL）在EX阶段完成PC值更新；在WB阶段将返回地址写入目的寄存器
- 其他状态与实验项目3中实现的多周期处理器状态机要求保持一致



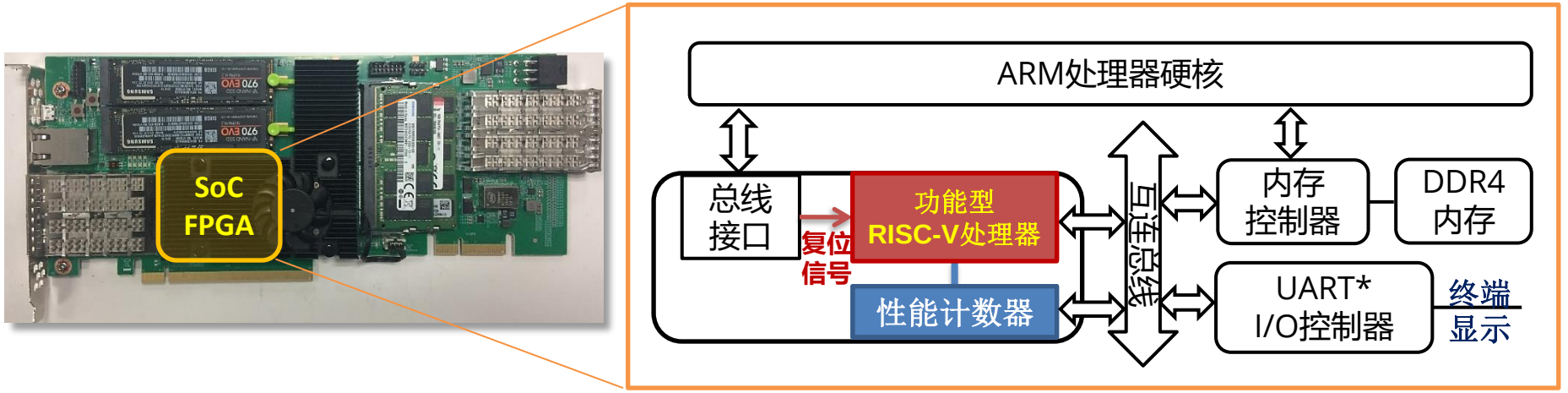
FPGA运行测试方法

❑在平台板卡的SoC-FPGA芯片中

a) 由ARM处理器向DDR4内存加载RISC-V处理器的各benchmark测试用例

b) ARM处理器释放RISC-V处理器核的复位信号，使RISC-V处理器开始运行

c) ARM处理器读取DDR4内存的0x0C地址，检查RISC-V是否正确运行测试用例



* 设计输入——设置实验流程的目标模块

- 同步框架更新

```
cd ~/COD-Lab && git pull upstream master
```

- 编辑脚本文件

```
cd ~/COD-Lab && vim lab_env.yml
```

```
variables:
  # TARGET_DESIGN can be "example", "alu", "reg_file",
  TARGET_DESIGN: "custom_cpu"

  # CPU_ISA must be either "mips" or "riscv32"
  CPU_ISA: "mips"
```

将mips改为riscv32

```
  # single_cycle or multi_cycle
  SIM_DUT_TYPE: "multi_cycle"
```

- 确认TARGET_DESIGN设置为custom_cpu
- 将CPU_ISA设置为riscv32（如右图）
- 确认SIM_DUT_TYPE设置为multi_cycle
- 将修改提交到代码仓库

```
cd ~/COD-Lab && git add lab_env.yml && git commit -m “lab_env: set design flow for RISC-V custom_cpu”
```

* 设计输入 —— RTL代码编写

- 添加多周期定制处理器数据及控制通路代码

```
cd ~/COD-Lab && vim fpga/design/ucas-cod/hardware/sources/custom_cpu/riscv32/custom_cpu.v
```

- 其他软硬件代码无需修改

```
module custom_cpu(
        input         clk,
        input         rst,

        //Instruction request channel
        output [31:0] PC,
        output        Inst_Req_Valid,
        input         Inst_Req_Ready,

        //Instruction response channel
        input  [31:0] Instruction,
        input         Inst_Valid,
        output        Inst_Ready,

        //Memory request channel
        output [31:0] Address,
        output        MemWrite,
        output [31:0] Write_data,
        output [ 3:0] Write_strb,
        output        MemRead,
        input         Mem_Req_Ready,

        //Memory data response channel
        input  [31:0] Read_data,
        input         Read_data_Valid,
        output        Read_data_Ready,

        input         intr,

        output [31:0] cpu_perf_cnt_0,
        output [31:0] cpu_perf_cnt_1,
        output [31:0] cpu_perf_cnt_2,
        output [31:0] cpu_perf_cnt_3,
        output [31:0] cpu_perf_cnt_4,
        output [31:0] cpu_perf_cnt_5,
        output [31:0] cpu_perf_cnt_6,
        output [31:0] cpu_perf_cnt_7,
        output [31:0] cpu_perf_cnt_8,
        output [31:0] cpu_perf_cnt_9,
        output [31:0] cpu_perf_cnt_10,
        output [31:0] cpu_perf_cnt_11,
        output [31:0] cpu_perf_cnt_12,
        output [31:0] cpu_perf_cnt_13,
        output [31:0] cpu_perf_cnt_14,
        output [31:0] cpu_perf_cnt_15
);
```

* 设计输入—— 将代码修改提交到本地仓库

❑ 每次修改代码后，都需要手动把修改提交（commit）到本地仓库

`cd ~/COD-Lab && git add` 已修改的文件路径

`cd ~/COD-Lab && git commit`（自行添加提交说明）

❑ 提交说明的编写要求请自学

- 尽量使用英文，并充分描述本次代码修改的目的、内容、预期达到的效果等信息
- 如何写好提交说明
  https://www.ruanyifeng.com/blog/2016/01/commit_message_change_log.html （中文参考）

* 设计输入—— 触发自动化开发流程（1）

- 在完成一次或多次代码修改及提交后，可推送代码到SERVE个人远程仓库

```
cd ~/COD-Lab && git push origin master
```

- 推送之后，自动化开发及部署流程就会在云平台开始运行
  - 可在浏览器中打开GitLab上的个人远程仓库，查看开发任务的执行情况
  - 调试方法与实验项目三一致

* 查看错误的行为仿真波形

在虚拟机中执行（注意：下面的命令是一条完整的命令，一行写不下所以出现换行）

```
cd ~/COD-Lab && make FPGA_PRJ=ucas-cod FPGA_BD=nf SIM_TARGET=custom_cpu
SIM_DUT=riscv32:multi_cycle WORKLOAD=simple_test:benchmark组名:benchmark名称 wav_chk
```

- 需要保证虚机可以上网
- 注意：命令第一行的最后有一个空格；不要按回车，直接输入第二行
- benchmark组名为：basic、medium、advanced、hello四个中的一个
- benchmark名称可根据SERVE网站上正在调试的仿真Job名称查看
- 波形文件的最后是出错指令的情况（与上页PPT输出信息对应）
- 从波形中找到出错点，结合测试用例的汇编指令序列，向前找到出错点
- 可按需在波形中添加要观察的信号

❑提交及验收说明

1* 在个人仓库中添加标签（十分重要）

- 在完成阶段I的所有设计和云上仿真、FPGA测试流程后，需在仓库中添加标签
- 便于助教老师后续进行代码评分
- 请执行如下命令（需要严格一致，否则影响成绩评分）

```
cd ~/COD-Lab && git tag -a custom_cpu-riscv32_multi_cycle -m “Release RISC-V multi_cycle custom CPU design”

cd ~/COD-Lab && git push origin master --tags
```

2* 实验报告撰写要求

- 描述定制RISC-V处理器的设计情况
  - 基本要求与实验项目3一致
- 实验报告大小尽量控制在5MB以内
- 实验报告请命名为“**prj4.pdf**”

3* 实验报告提交及远程推送方法

- **cd ~/COD-Lab && mkdir -p reports**
- **将prj4.pdf这个文件拷贝到~/COD-Lab/reports**
  - 本地虚拟机：通过图形界面拷贝
  - 云端虚拟机：通过SFTP上传（具体需查看自己使用terminal终端的SFTP使用方法）
- **提交到个人本地仓库**
  cd ~/COD-Lab && git add reports/ && git commit -m “docs: add prj4 report”
- **推送到个人远程仓库**
  cd ~/COD-Lab && git push origin master
- **请确保提交报告前，实验编写的Verilog HDL代码已完成云端测试**

4. 课堂验收要求（5月6日开始）

1. **实验态度端正，完成代码编写和上板测试，可得基准成绩50分**
   - 查看run/log目录下的五组benchmark运行结果
2. **其他检查要点 （满分40分）**
   - 状态机编码用one-hot，并按照三段式正确表述（+10分）
   - 除状态机第二段外的组合逻辑必须用assign语句描述（+10分）
   - RISC-V/MIPS指令集性能分析对比（+20分）
3. **同学对代码及实验内容的思考（满分10分，助教根据沟通情况打分）**
4. **同学需根据助教建议修改代码，并在实验报告中进行说明。助教会根据课堂验收记录，检查最终提交代码修改情况**
5. **如果明显抄袭 0分记录（对代码完全说不清楚的，疑似抄袭）**

Q & A ?


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