# Computer Architecture Lab

Xiaohong Jiang

College of Comuter Science

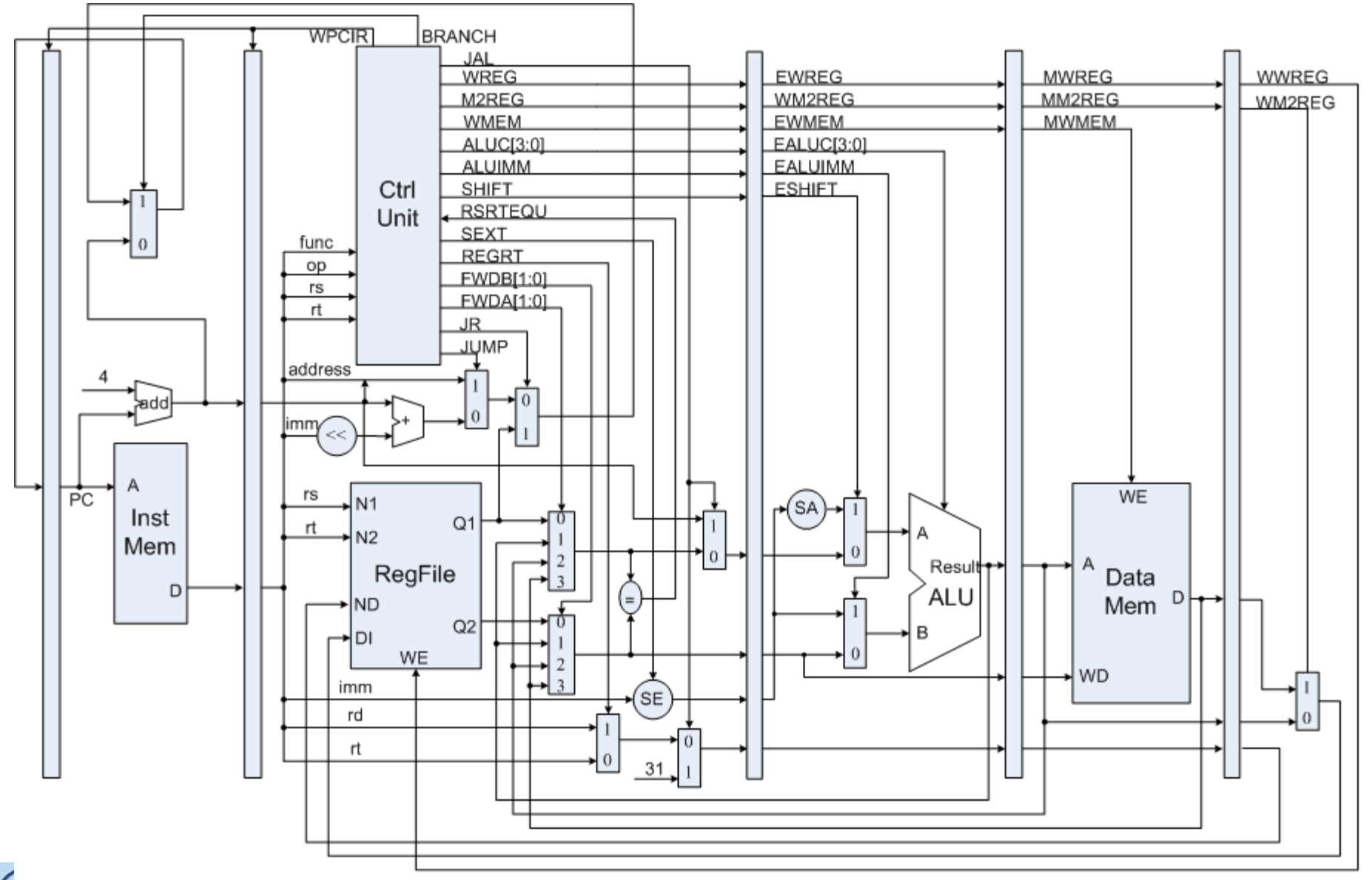
# Prerequisite

- Lab for Fundamentals of Logic and Computer Design
- Lab for Organization
- Lab environment：
  - FPGA board：Xilinx Spartan-3E
  - Software：Xilinx ISE 10.1i
  - HDL：Verilog

# Lab Objective

- Learn the operation of Spartan3E Board and the usage of ISE.
- Understand the principle of the pipelined CPU and MIPS instructions.
- Design the pipelined CPU that can execute 31 MIPS instructions correctly on Spartan3E board step by step according to the project tutorial (step-over-lab is not allowed, ex doing lab2/lab3 together).

# Data path and control unit（Partial）



# Schedule

- Lab 1). Warmup Run you single-cycle CPU on 3E board and try to add instructions to support 15 MIPS instructions.
- Lab 2). 5-stage pipelined CPU with 15 MIPS instructions (only required to execute in pipeline).
- Lab 3). Implementing "stall" when have hazards so that CPU can execute program correctly.
- Lab 4). Implementing “forwarding paths” and “predict-not-taken” to make CPU run faster.
- Lab 5). The whole CPU with 31 instructions. Adding the additional 16 instrutions to enable your pipelined CPU to support 31 MIPS.

# 15 common used MIPS instructions

| MIPS Instructions | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | [31..26] | [25..21] | [20..16] | [15..11] | [10..06] | [05..00] | Operations | |
| R-type | op | rs | rt | rd | sa | func | | |
| add | 000000 | rs | rt | rd | 00000 | 100000 | rd <-- rs + rt; | PC <-- PC + 4 |
| sub | 000000 | rs | rt | rd | 00000 | 100010 | rd <-- rs - rt; | PC <-- PC + 4 |
| and | 000000 | rs | rt | rd | 00000 | 100100 | rd <-- rs & rt; | PC <-- PC + 4 |
| or | 000000 | rs | rt | rd | 00000 | 100101 | rd <-- rs \| rt; | PC <-- PC + 4 |
| sll | 000000 | 00000 | rt | rd | sa | 000000 | rd <-- rt << sa; | PC <-- PC + 4 |
| srl | 000000 | 00000 | rt | rd | sa | 000010 | rd <-- rt >> sa (logical); | PC <-- PC + 4 |
| sra | 000000 | 00000 | rt | rd | sa | 000011 | rd <-- rt >> sa (arithmetic); | PC <-- PC + 4 |
| I-type | op | rs | rt | immediate | | | | |
| addi | 001000 | rs | rt | immediate | | | rt <-- rs + (sign_extend)immediate; | PC <-- PC + 4 |
| andi | 001100 | rs | rt | immediate | | | rt <-- rs & (zero_extend)immediate; | PC <-- PC + 4 |
| ori | 001101 | rs | rt | immediate | | | rt <-- rs \| (zero_extend)immediate; | PC <-- PC + 4 |
| lw | 100011 | rs | rt | immediate | | | rt <-- memory[rs + (sign_extend)immediate]; | PC <-- PC + 4 |
| sw | 101011 | rs | rt | immediate | | | memory[rs + (sign_extend)immediate] <-- rt; | PC <-- PC + 4 |
| beq | 000100 | rs | rt | immediate | | | if (rs == rt) PC <-- PC + 4 + (sign_extend)immediate<<2; else | PC <-- PC + 4 |
| bne | 000101 | rs | rt | immediate | | | if (rs != rt) PC <-- PC + 4 + (sign_extend)immediate<<2; else | PC <-- PC + 4 |
| J-type | op | address | | | | | | |
| j | 000010 | address | | | | | PC <-- (PC+4)[31..28],address<<2 | |

# Pipelined CPU supporting execution of 31 MIPS instructions

| MIPS Instructions | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | **31..26** | **25..21** | **20..16** | **15..11** | **10..6** | **5..0** | **Operations** | |
| **R-type** | op | rs | rt | rd | sa | func | | |
| **add** | | rs | rt | rd | 00000 | 100000 | rd = rs + rt; with overflow | PC+=4 |
| **addu** | | rs | rt | rd | 00000 | 100001 | rd = rs + rt; without overflow | PC+=4 |
| **sub** | | rs | rt | rd | 00000 | 100010 | rd = rs - rt; with overflow | PC+=4 |
| **subu** | | rs | rt | rd | 00000 | 100011 | rd = rs - rt; without overflow | PC+=4 |
| **and** | | rs | rt | rd | 00000 | 100100 | rd = rs & rt; | PC+=4 |
| **or** | | rs | rt | rd | 00000 | 100101 | rd = rs \| rt; | PC+=4 |
| **xor** | | rs | rt | rd | 00000 | 100110 | rd = rs ^ rt; | PC+=4 |
| **nor** | | rs | rt | rd | 00000 | 100111 | rd = ~(rs \| rt); | PC+=4 |
| **slt** | 000000 | rs | rt | rd | 00000 | 101010 | if(rs < rt)rd = 1; else rd = 0; <(signed) | PC+=4 |
| **sltu** | | rs | rt | rd | 00000 | 101011 | if(rs < rt)rd = 1; else rd = 0; <(unsigned) | PC+=4 |
| **sll** | | 00000 | rt | rd | sa | 000000 | rd = rt << sa; | PC+=4 |
| **srl** | | 00000 | rt | rd | sa | 000010 | rd = rt >> sa (logical); | PC+=4 |
| **sra** | | 00000 | rt | rd | sa | 000011 | rd = rt >> sa (arithmetic); | PC+=4 |
| **sllv** | | rs | rt | rd | 00000 | 000100 | rd = rt << rs; | PC+=4 |
| **srlv** | | rs | rt | rd | 00000 | 000110 | rd = rt >> rs (logical); | PC+=4 |
| **srav** | | rs | rt | rd | 00000 | 000111 | rd = rt >> rs(arithmetic); | PC+=4 |
| **jr** | | rs | 00000 | 00000 | 00000 | 001000 | | PC=rs |

# Pipelined CPU supporting execution of 31 MIPS instructions

| MIPS Instructions | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | **31..26** | **25..21** | **20..16** | **15..11** | **10..6** | **5..0** | **Operations** | |
| **I-type** | op | rs | rt | immediate | | | | |
| **addi** | 001000 | rs | rt | imm | | | rt = rs + (sign_extend)imm; with overflow | PC+=4 |
| **addiu** | 001001 | rs | rt | imm | | | rt = rs + (sign_extend)imm;without overflow | PC+=4 |
| **andi** | 001100 | rs | rt | imm | | | rt = rs & (zero_extend)imm; | PC+=4 |
| **ori** | 001101 | rs | rt | imm | | | rt = rs \| (zero_extend)imm; | PC+=4 |
| **xori** | 001110 | rs | rt | imm | | | rt = rs ^ (zero_extend)imm; | PC+=4 |
| **lui** | 001111 | 00000 | rt | imm | | | rt = imm << 16; | PC+=4 |
| **lw** | 100011 | rs | rt | imm | | | rt = memory[rs + (sign_extend)imm]; | PC+=4 |
| **sw** | 101011 | rs | rt | imm | | | memory[rs + (sign_extend)imm] <-- rt; | PC+=4 |
| **beq** | 000100 | rs | rt | imm | | | if (rs == rt) PC+=4 + (sign_extend)imm <<2; | PC+=4 |
| **bne** | 000101 | rs | rt | imm | | | if (rs != rt) PC+=4 + (sign_extend)imm <<2; | PC+=4 |
| **slti** | 001010 | rs | rt | imm | | | if (rs < (sign_extend)imm) rt =1 else rt = 0; less than signed | PC+=4 |
| **sltiu** | 001011 | rs | rt | imm | | | if (rs < (zero_extend)imm) rt =1 else rt = 0; less than unsigned | PC+=4 |
| **J-type** | op | address | | | | | | |
| **j** | 000010 | address | | | | | PC = (PC+4)[31..28],address<<2 | |
| **jal** | 000011 | address | | | | | PC = (PC+4)[31..28],address<<2 ; $31 = PC+4 | |

# Grading 32% in all

- Participation 4%
- Lab1-5: 6%, 4%, 5%, 8%, 5%
- 5 Lab reports
  - Lab result 60%, report 40%

- Alternative Labs (32%） + a bonus 10%
  - Implement a pipelined LC3
  - 16%, 7%, 7%, 8%

# How to do Lab ?

- You are highly encouraged to do the lab assignment in group of 2 students, but you need to write and submit your lab report all by yourself.

- Lab report template will be uploaded to the course website.

# Lab report submission:

- Submit your lab report to the course website into the lab directory naming in StID_name_lab1.doc, …, StID_name_lab4.doc, and **StID_name_lab5.rar** including StID_name_lab5.doc **and your lab work directory**.

- Submission deadline will be announced on course website.

# Materials

1、ftp://archlab10:archlab10@10.71.75.200

User guide of Spartan-3E board

Xilinx ISE_10.1.iso

2、《Instruction for lab of computer architecture》in Chinese

（You can get it from textbook center of Zhejiang Univ.）