CG3207 Lab 2
Basic CPU Design

Neil Banerjee

neil@nus.edu.sg

Lab 2 Objectives

ARM Version

- Implement the basic ARMv3 processor seen in lectures
  - LDR, STR w/+ve imm offset
  - AND, OR, ADD, SUB w/Src2 as immediate or register, no shifts
  - B
- Add the following features:
  - CMP, CMN
  - LDR, STR
  - Register w/imm shift for DP insts

RISC-V Version

- Implement the following basic instructions
  - add, addi, sub, and, andi, or, ori
  - lw, sw
  - beq, bne, jal (no linking)
- Add the following features:
  - lui, auipc
  - sll, srl, sra

File Hierarchy

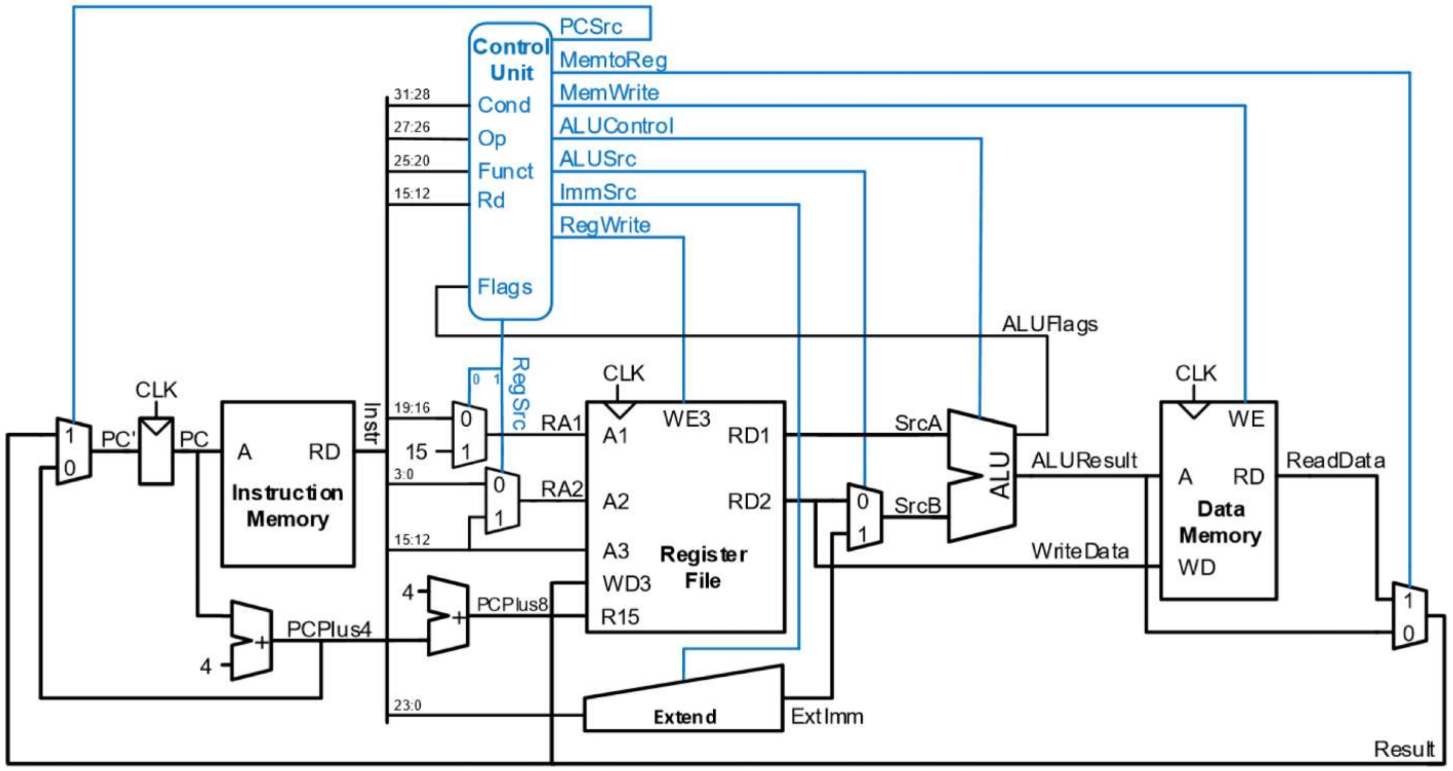
ARMv3

- TOP
    - Wrapper
        - ARM
            - ALU
            - CondLogic
            - Decoder
            - Extend
            - ProgramCounter
            - RegFile
            - Shifter

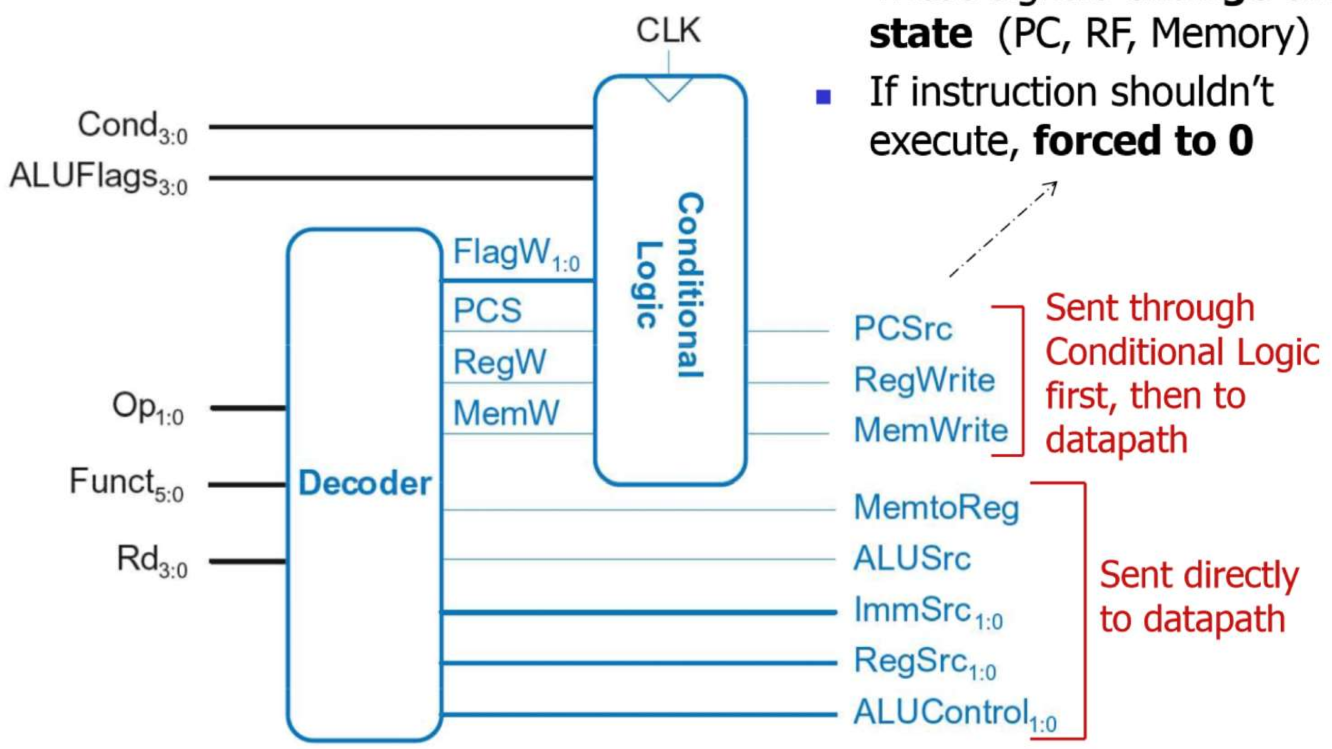
RISC-V

- TOP
    - Wrapper
        - RV
            - ALU
                - Shifter
            - CondLogic
            - Decoder
            - Extend
            - ProgramCounter
            - RegFile

Block Design (ARM overview)

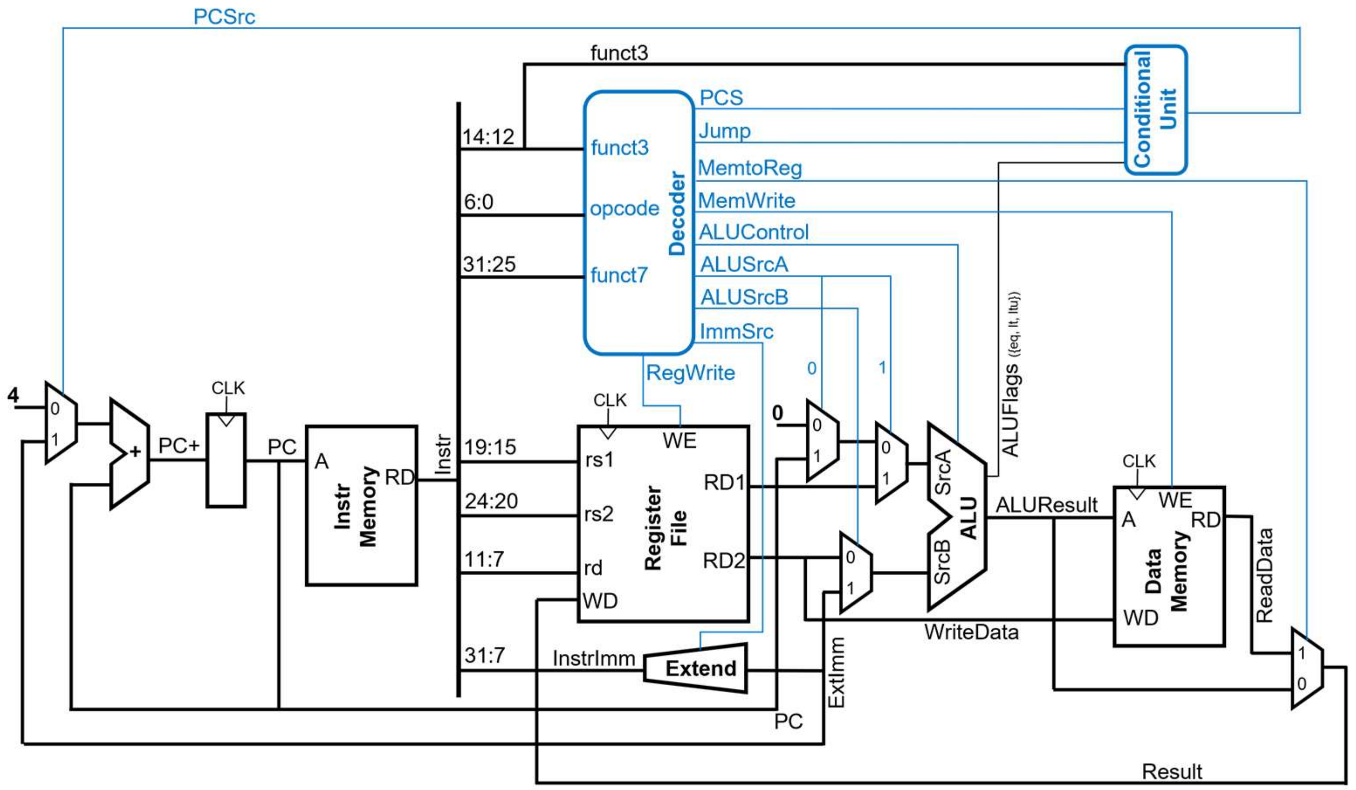


Block Design (ARM Control Unit)



- These signals **change the state** (PC, RF, Memory)
- If instruction shouldn't execute, **forced to 0**

Block Design (RISC-V)



Memory Map

ARM

Address	Attributes	Description
0x0-0x1FC	R	Instruction ROM
0x200-0x3FC	R	Data ROM
0x800-0x9FC	RW	Data RAM
0xA00-0xBFC	X	Unused
0xC00	W	LED[15:8]
0xC04	R	DIP switches
0xC08	R	Push buttons
0xC0C	RW	UART in/out
0xC10	R	CONSOLE_IN_valid
0xC14	R	CONSOLE_OUT_ready
0xC18	W	7-segment display

RISC-V

Address	Attributes	Description
0x0-0x1FC	R	Instruction ROM
0x2000-0x21FC	R	Data ROM
0x2200-0x23FC	RW	Data RAM
0x2400	W	LED[15:8]
0x2404	R	DIP switches
0x2408	R	Push buttons
0x240C	RW	UART in/out
0x2410	R	CONSOLE_IN_valid
0x2414	R	CONSOLE_OUT_ready
0x2418	W	7-segment display

Setting up the project

- Download Lab 2 Template from Wiki
  - Download correct file for ARM/RV as needed
- Create a new project in Vivado, add sources to project
  - I recommend copying sources into project so you can use git
  - Use Verilog/VHDL files depending on your preference
  - Remember to import the correct constraints file (.xdc)
- Read wiki carefully
- Synthesize blocks to ensure no synthesis errors, then simulate
- Restart != Relaunch for sim – must relaunch if you change HDL

Testing options

- Use sample assembly file from Lab 1
  - May need to modify it
- Use HelloWorld assembly file provided
  - Not comprehensive, not simple – just an example
  - Use RealTerm for UART
- Write your own assembly code
  - Best option, recommended for demonstration
  - Demonstrate all the implemented features
  - No need to use all the hardware features – as long as it is convincing
- For option 1, 3 – modify test_Wrapper file as necessary

Submission details

- Demonstrate in Week 7
- One single ASM program to demonstrate all features is best
- Upload single archive Lab2_Monday<nn>.zip (e.g. Lab2_Monday42.zip)
    - Created/modified .v/.vhd files
    - .bit files
    - .asm files
    - readme.txt with purpose of each file created/modified
    - Files MUST be exact same as used for demonstration
    - Do not zip up complete Vivado project
- One submission per group, last submission from any group member is final
- Extra features, performance improvements NOT graded (leave for Lab 4!)
- Testbench AND hardware simulation needed