Logic Design Laboratory

Final project announcement
May 21st, 2019

Prof. Jae W. Lee (jaewlee@snu.ac.kr)
Architecture and Code Optimization (ARC) Lab.
Seoul National University

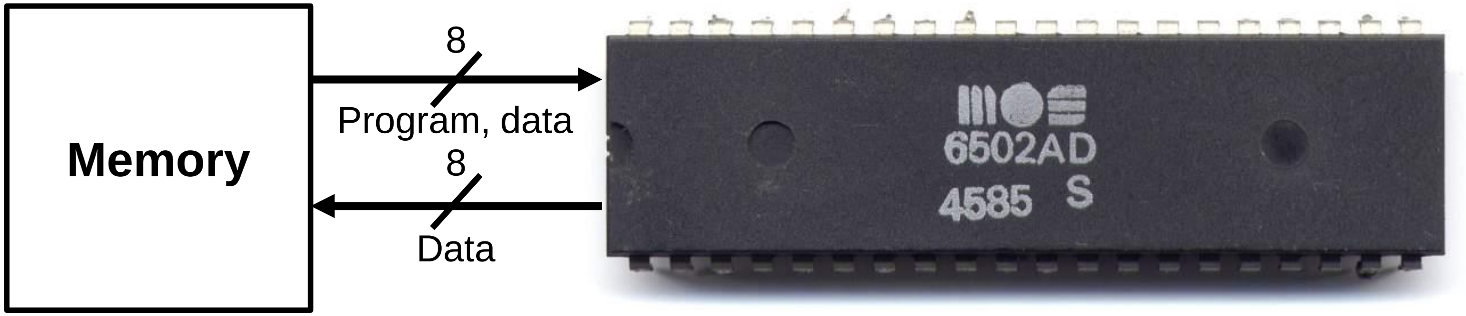
TA: Jeonghun Gong, Yeonhong Park
Myeongseok Seong, Sunmin Jeong

Contents

- **Introduction to simple 8-bit microprocessor**
- **Project Overview**
- **Architecture**
  - Data Path
  - ISA(Instruction Set Architecture)
- **Implementation Guide**
- **Grading Policy**
- **Submission Guide**

Introduction to 8-bit microprocessor

- **What is 8-bit microprocessor?**
  - It reads program from memory and operates as it directs.
  - Data is also stored on and loaded from memory.
  - Basic data size is 8-bit.
    - It reads, processes and writes 8-bit data.
    - It has several 8-bit registers.



MOS Technology 6502 microprocessor (1975)

Introduction to 8-bit microprocessor

- **Your microprocessor reads, decodes program and execute it.**
  - Program is a sequence of instructions
  - Instruction contains all the information about what should be done by microprocessor

Machine code* **(Instruction)**

01000100
01100100
11010000
11010000
11010010
10000100
…

Decode, and Execute!

What is done inside

r0 = r1[0]
r2 = r1[0]
r0 = r0 + r0
r0 = r0 + r0
r0 = r0 + r2
r1[0] = r0
…

*Not actual MOS 6502 machine code

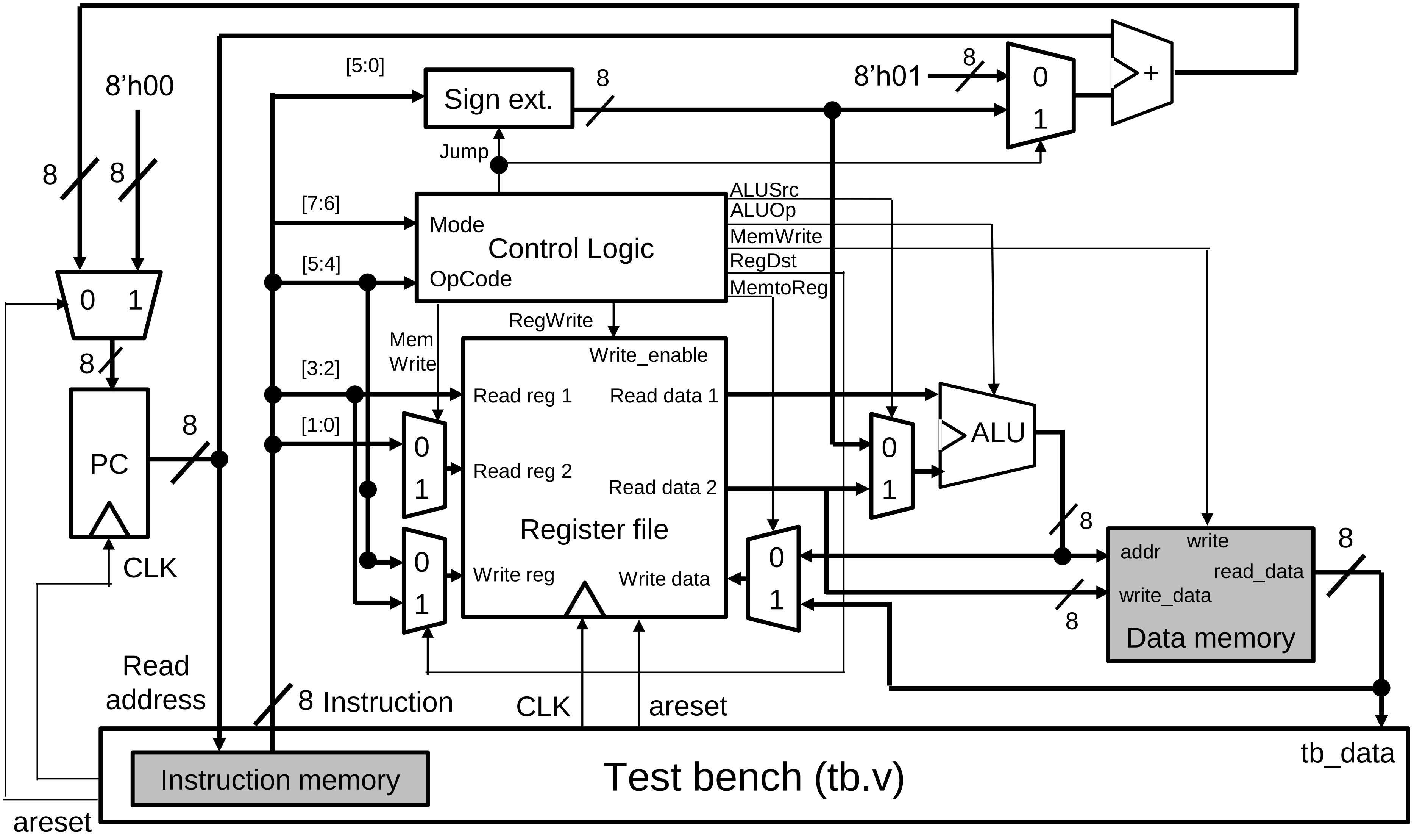
Project Overview

- Implement a Simple Microprocessor in Verilog and program it in FPGA. Mimic a real computer.
- Implement each component as a module in Verilog. You should make Register, ALU, Control unit, Program Counter, and the other components into separate modules.
- Connect each module in a proper way using bus.
- This is an **individual project**. You **must not** share your code with your partner. Your partnership is officially over!

Project Overview

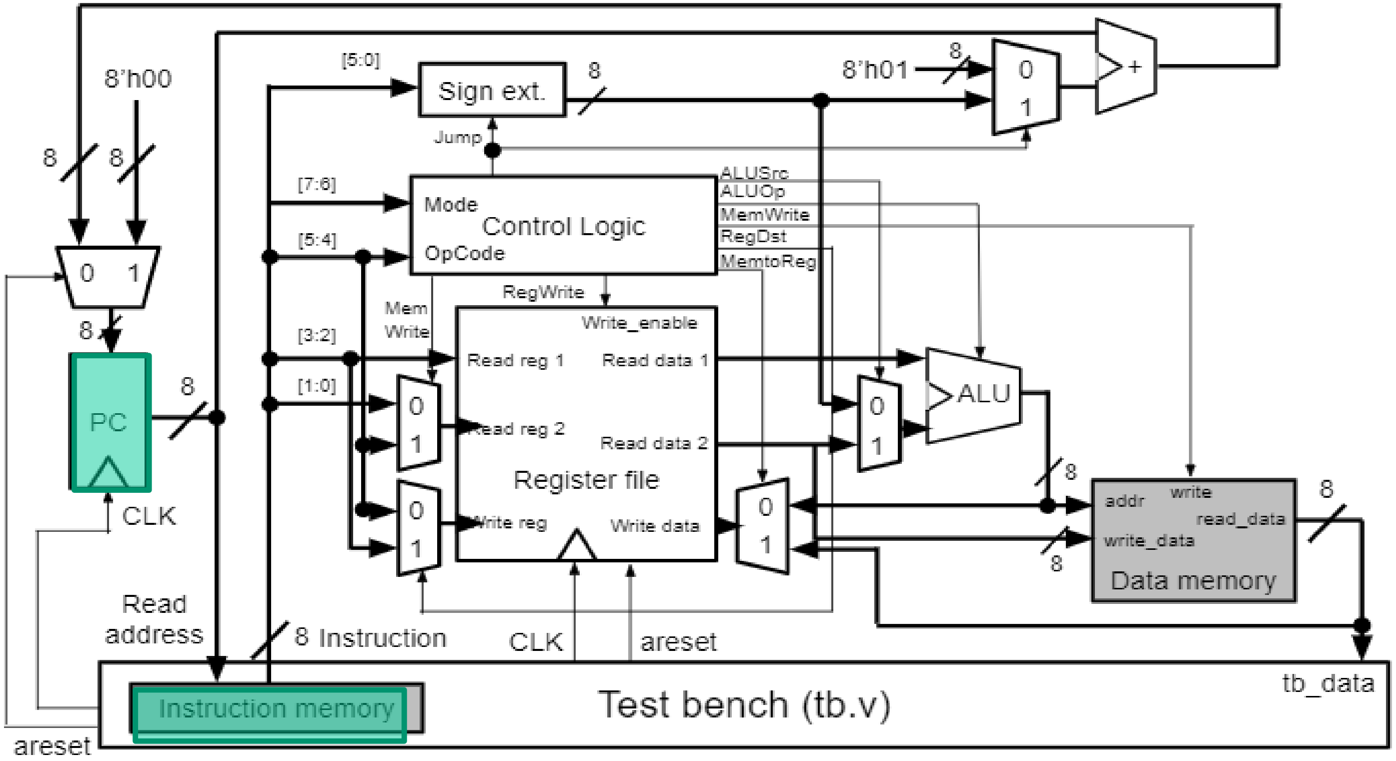
- 8-bit Microprocessor
  - Instruction size: 8-bit
  - Register & Data size : 8-bit
- # of instructions : 6 (add, sub load, store, jump, nop) + 1 (addi)
- # of registers : 4
- Grading target : Final state of data memory
  - Data memory values (total #: 32) after test sequence of instructions will be compared to the correct answers precalculated by TAs.

Architecture: Data Path



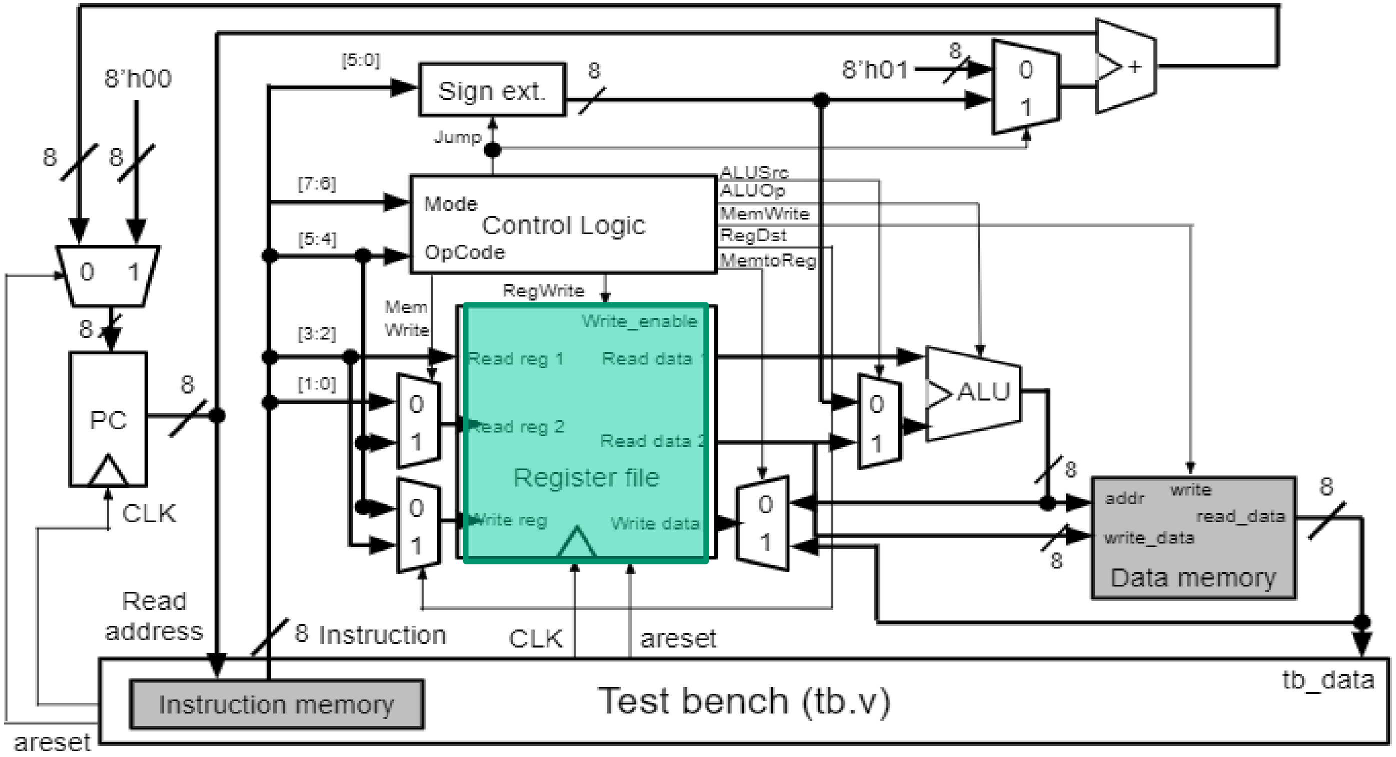
Architecture: PC

- **It should have special register "PC (program counter)"**
  - PC contains the address of current instruction to fetch from instruction memory, and increment the address by one unless there is a control transfer instruction ('jump' in this architecture).



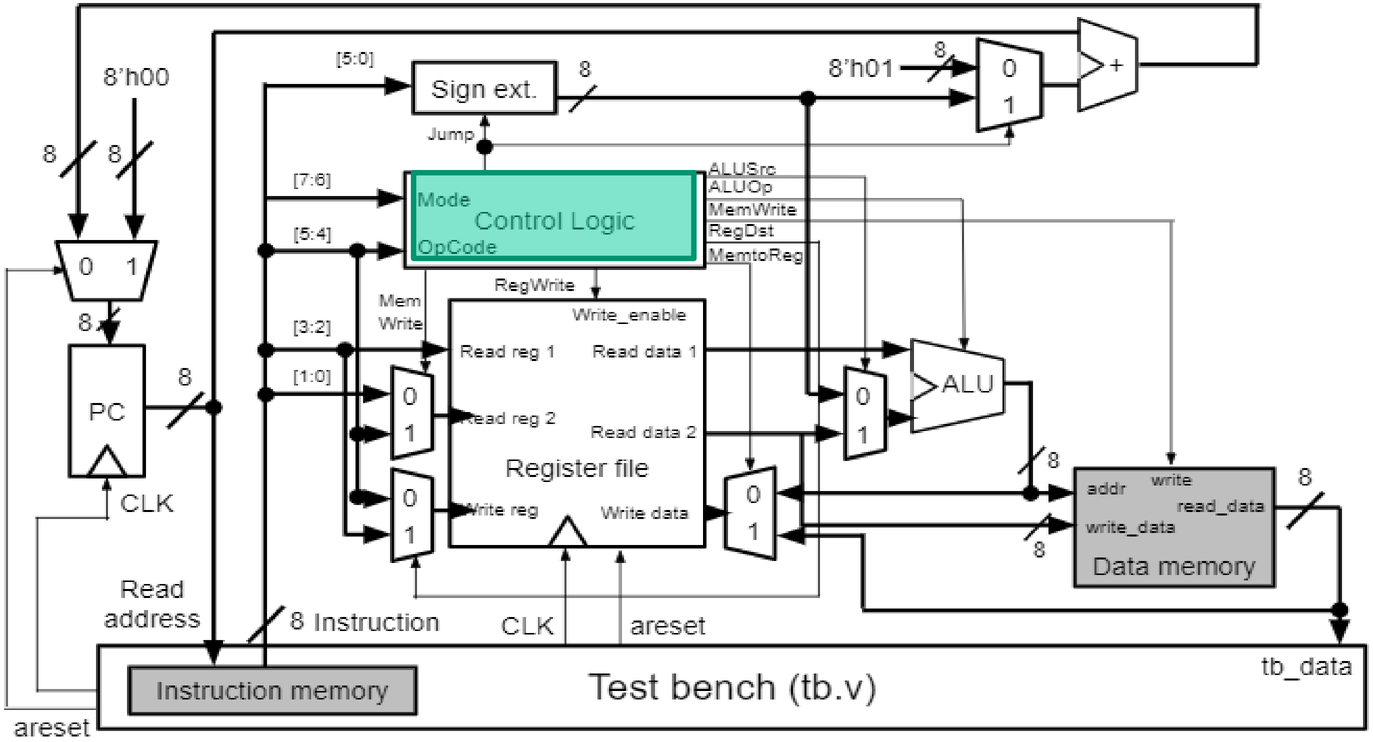
Architecture: Register File

- **It has four registers (r0 to r3) in the register file.**
  - Register to be read/written is selected by “read reg/write reg” signals. It has two read port and one write port.
  - Register value is updated if write enable is on.



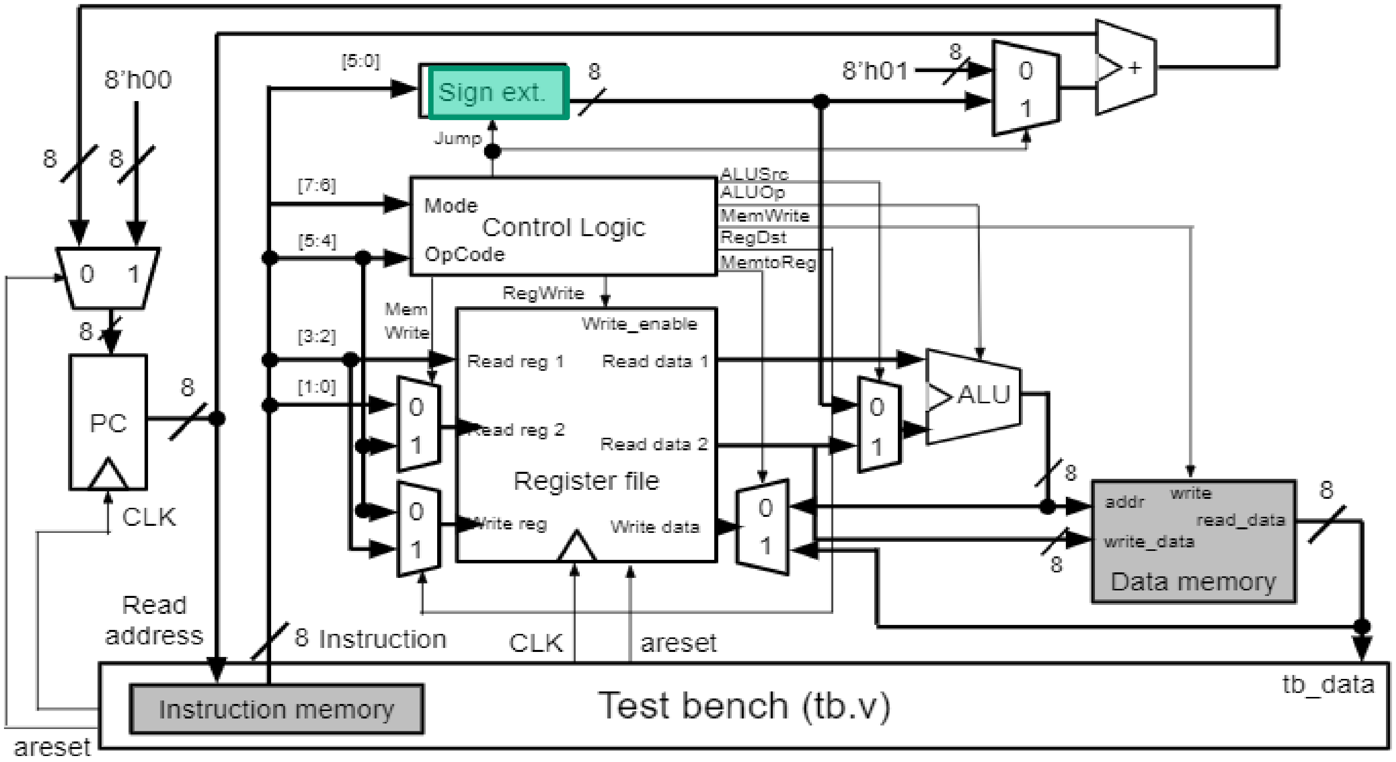
Architecture: Control Logic

- **Control logic**
  - Control logic decodes instruction and sends out control signals across the microprocessor.



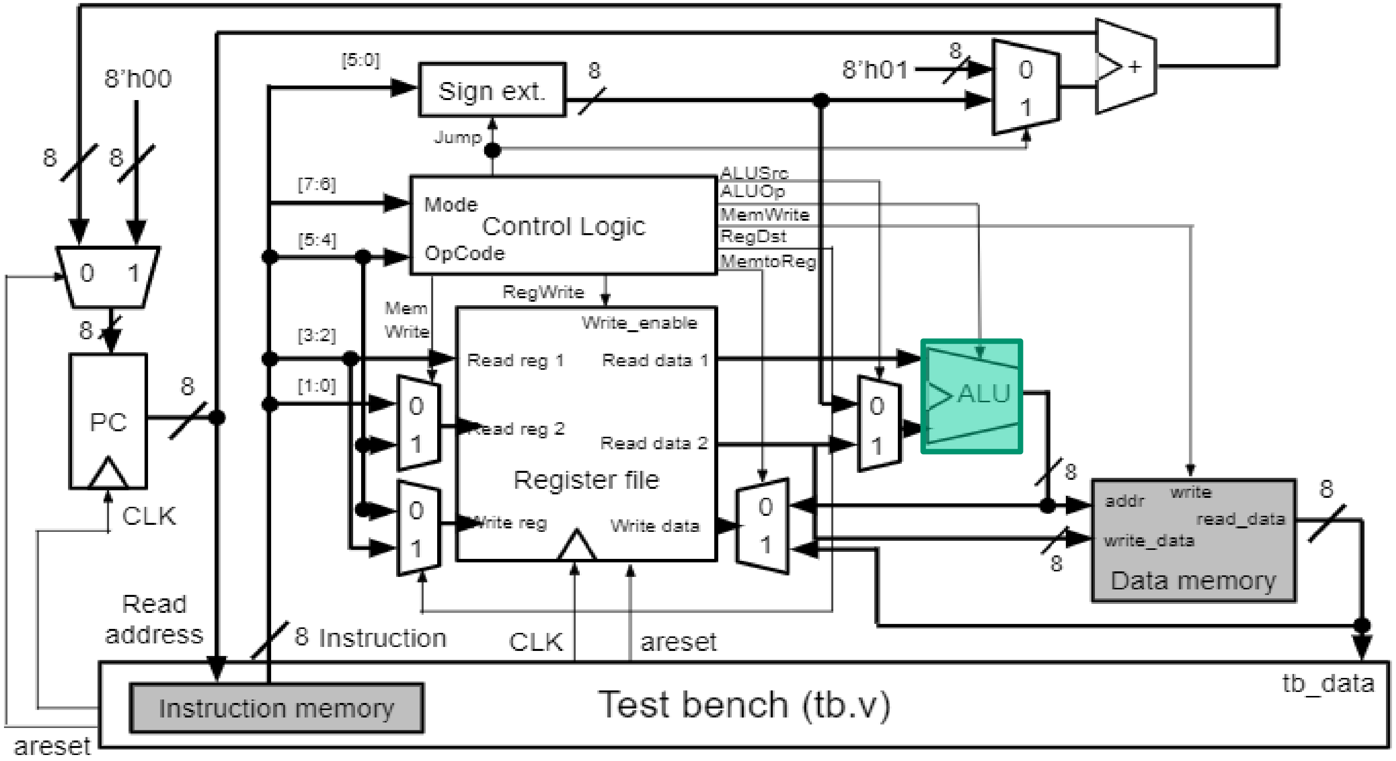
Architecture: Sign Extension Unit

- **Sign extension unit.**
  - Used for sign extension of [off] field in load, store and jump instructions.



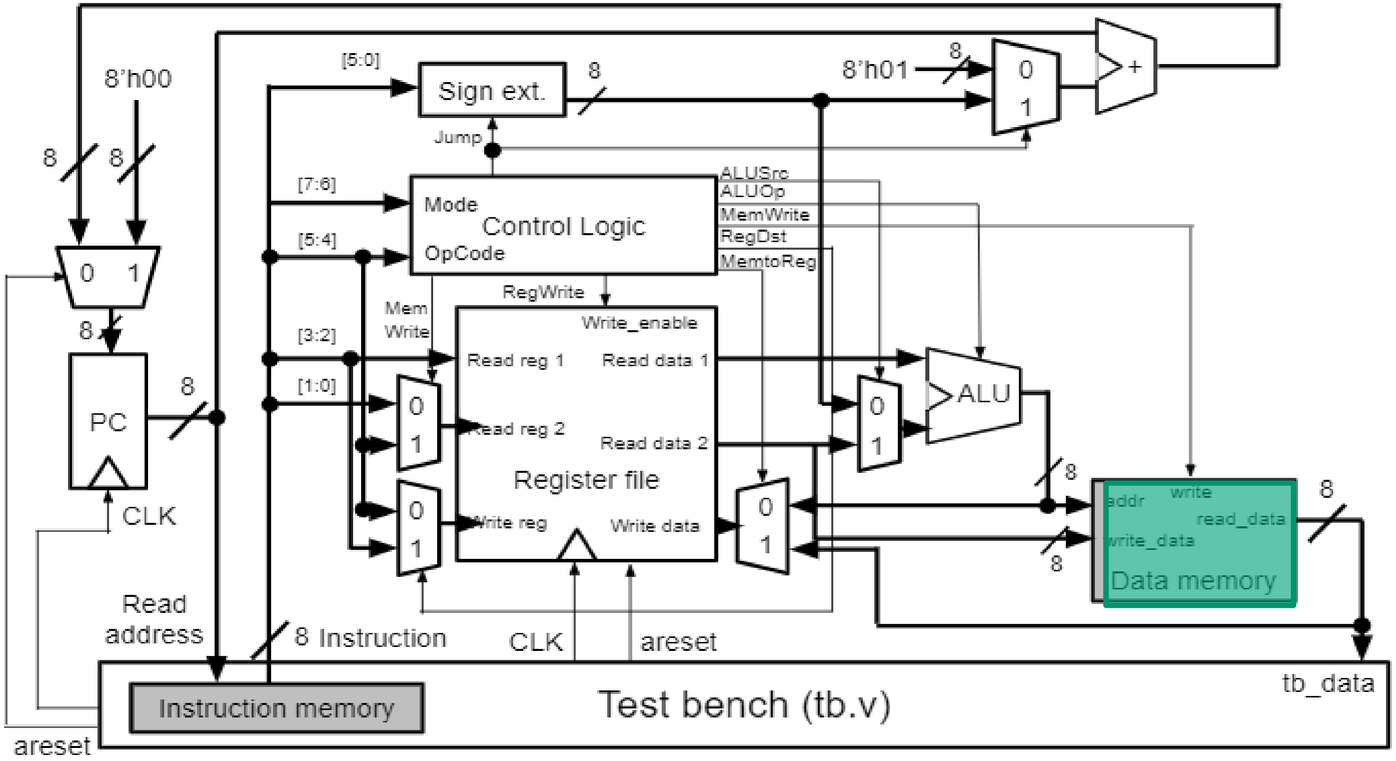
Architecture: ALU

- **Arithmetic Logic Unit**
  - ALU receives two inputs and execute arithmetic operation based on the control signal.



Architecture: Data Memory

- **Data memory (given)**
  - You can write to or read from certain address in data memory.
  - Set 'write' bit high when writing. (low when reading)
    - Your data will be updated at rising edge of clk.



Architecture: ISA

- **ISA(Instruction Set Architecture)**
  - ISA is a set of the instructions that a processor understands. It specifies how each binary code should be interpreted. It is an interface between software and hardware
  - In this microprocessor, each instruction is 8-bit and 6 different instructions should be handled.

    (Add / Sub / Load/ Store / Jump / NOP)

Architecture: ISA

Type	Opcode	Description	Note
Add	11 01 [rd] [rs]	$rd = $rd +$ rs	
Sub	11 10 [rd] [rs]	$rd = $rd – $rs	
Load	01 [rt] [rs] [off]	$rt = mem[$rs+off]	'off' is sign extended
Store	10 [rs] [rd] [off]	mem[$rd+off] = $rs	'off' is sign extended
Jump*	00 [off (6 bits)]*	$PC = $PC + off	'off' is sign extended
NOP	11 00 XX XX	Do nothing	

Reg. no	Reg. code
r0	00
r1	01
r2	10
r3	11

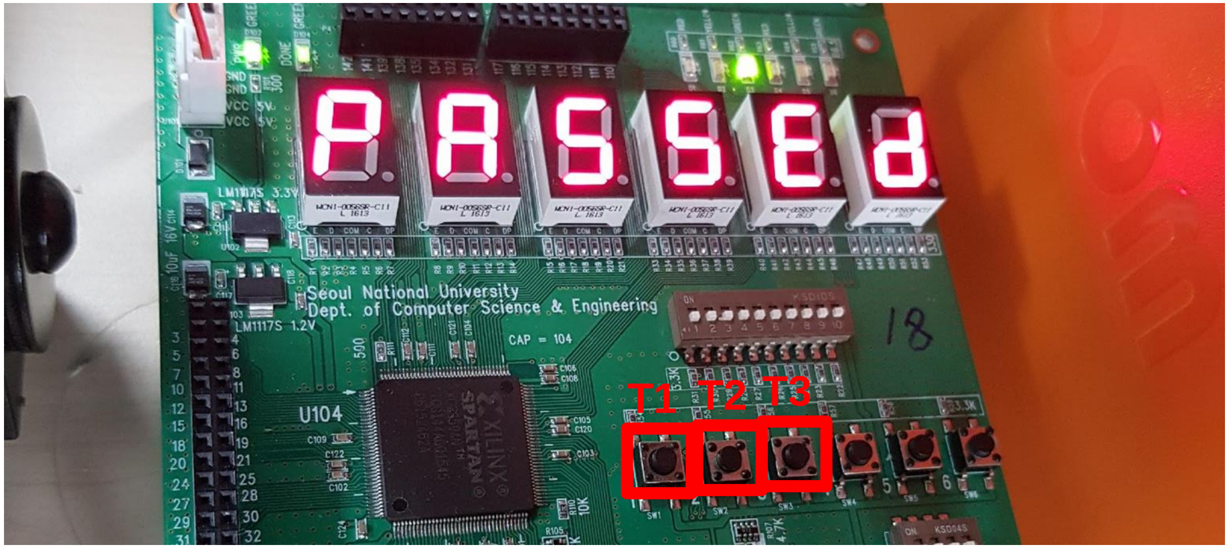
* Instructions other than Jump should increment PC by 1.
* […] field is 2 bits except for jump operation.

Implementing Guide

- **Implement your design based on ‘cpu.v’**
  - Implement all the components previously explained (except data memory).
  - Make sure not to modify port declaration.
  - Read comments carefully for further instructions.

Implementing Guide: Test by yourself!

- **Make a project with given tb.v and your cpu.v**
  - Program it on the board.
  - Refer to tb.v:[5-25] for pin mapping.
- **tb.v has three test cases.**
  - Each tactile switch corresponds to certain test case.
  - Seven segment displays will show result.


PASSEd
Seoul National University
Dept. of Computer Science & Engineering
T1
T2
T3

Bonus stage: Expanded instruction set

- **Addi (Add immediate) instruction**
  - You will get extra credit (10% of final project credit) for this.
  - If your final project score goes over 100%, the surplus will be added to your lab score (not overall course score).

Type	Opcode	Description	Note
Add	11 01 [rd] [rs]	$rd = $rd +$ rs	
Sub	11 10 [rd] [rs]	$rd = $rd – $rs	
Load	01 [rt] [rs] [off]	$rt = mem[$rs+off]	‘off’ is sign extended
Store	10 [rs] [rd] [off]	mem[$rd+off] = $rs	‘off’ is sign extended
Jump*	00 [off (6 bits)]*	$PC = $PC + off	‘off’ is sign extended
NOP	11 00 XX XX	Do nothing	
Addi (Add immediate)	11 11 [rd] [imm]	$rd = $rd + imm	‘imm’ is sign extended ‘imm’ is 2s complement

Grading Policy

- Report (30%)
  - Explain the overall design and structure in detail
  - Specify the functionality of each module in your implementation
  - Verify your implementation is correct with simulation result
  - Around 5 pages recommended
- Completeness (70%)
  - Six instruction sequences will be used for grading. Three of them are in tb.v and the other three will be more advanced ones.

    (test 0/1/2: 7 points, test 3/4: 14 points, test 5: 21 points)
- Extra credits (10%)
  - Refer to the previous slide

Term project accounts for 50% of the total lab score!

Submission Guide

- What to submit
  - CPU source code (student_id.v (Example: 2019-12345.v))
    - You may use multiple files during implementation, but please merge them into a single v file when you submit it.
    - tb.v is only for self-testing. You do not have to submit.
  - Report (student_id.pdf (Example: 2019-12345.pdf))
- Due: Before 2019. 06. 11 (Tue) 23:59
  - No delay. Late submission is not allowed.
- Please submit source code and report separately. Do not compress.