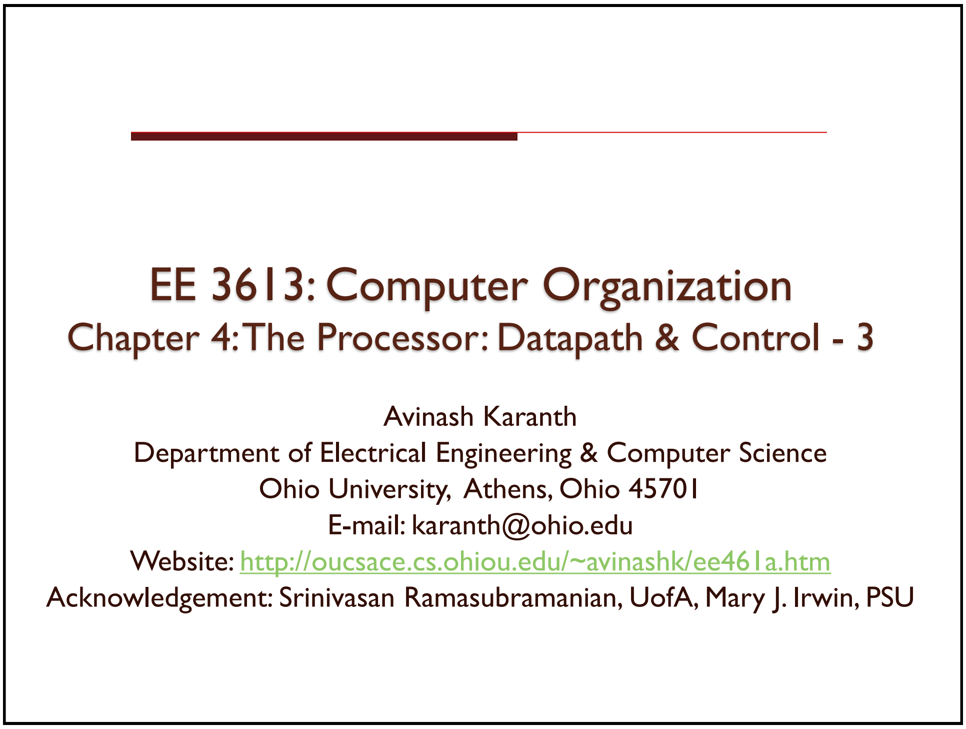


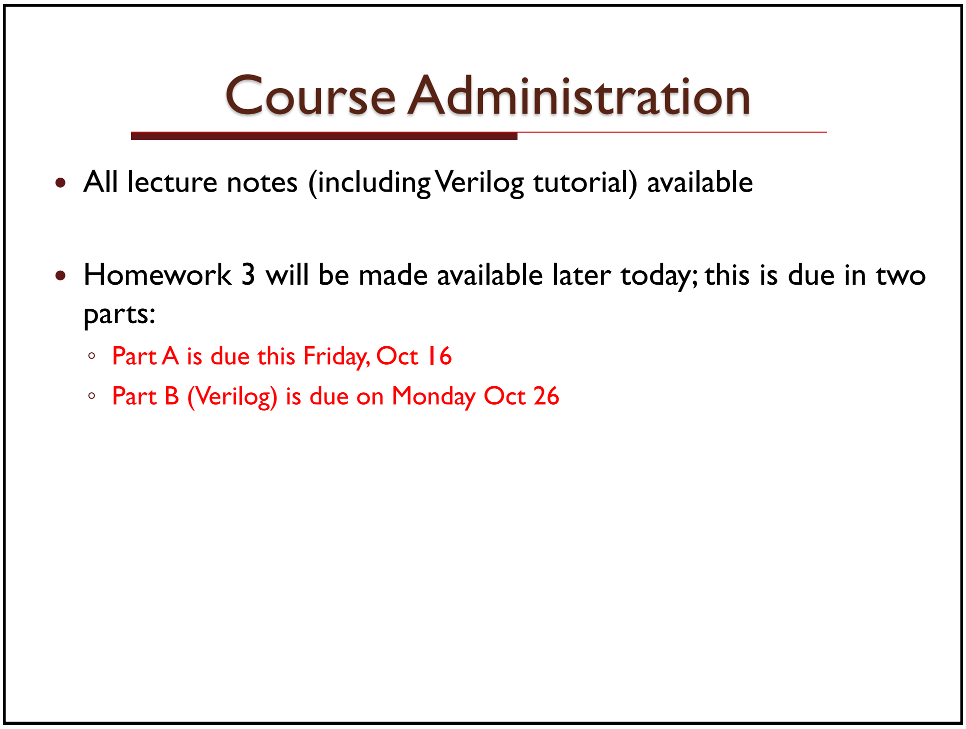
EE 3613: Computer Organization

Chapter 4: The Processor: Datapath & Control - 3

Avinash Karanth
Department of Electrical Engineering & Computer Science
Ohio University, Athens, Ohio 45701
E-mail: karanth@ohio.edu
Website: http://oucsace.cs.ohiou.edu/~avinashk/ee461a.htm

Acknowledgement: Srinivasan Ramasubramanian, UofA, Mary J. Irwin, PSU

1



Course Administration

- All lecture notes (including Verilog tutorial) available
- Homework 3 will be made available later today; this is due in two parts:
  - Part A is due this Friday, Oct 16
  - Part B (Verilog) is due on Monday Oct 26

2

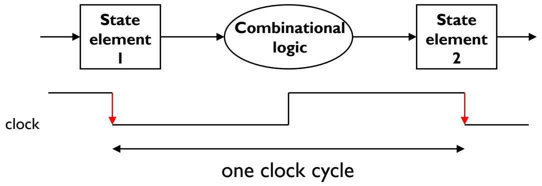
Datapath and Control

- We will design a simplified MIPS processor
- The instructions supported are
  - Memory-reference instructions: lw, sw
  - Arithmetic-logical instructions: add, sub, and, or, slt
  - Control-flow instructions: beq, j
- Generic Implementation
  - Use the program counter (PC) to supply the instruction address and fetch the instruction from memory (and update the PC)
  - Decode the instruction (and read the registers)
  - Execute the instruction
- All instructions (except j) use the ALU after reading the registers
  - How? Memory-reference ? Arithmetic ? Control-flow ?

3

Clocking Methodologies

- The clocking methodology defines when signals can be read and when they can be written
  - An edge triggered methodology
- Typical execution
  - Read contents of state elements
  - Send values through combinational logic
  - Write results to one or more state elements

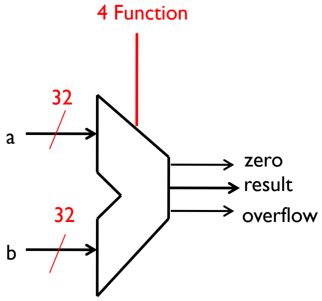


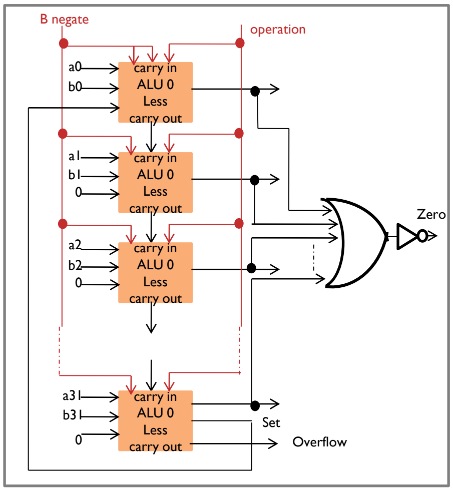
- Assumes that state elements are written on every clock cycle; if not, need explicit write control signal
  - Writes occur only when both the write control is asserted and clock edge occurs

4

What blocks do we need?

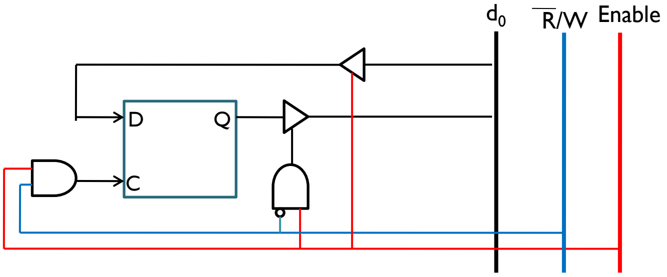
- We need an ALU – already designed a 4 function (ADD/SUB, OR, AND, SLT) and BRANCH





5

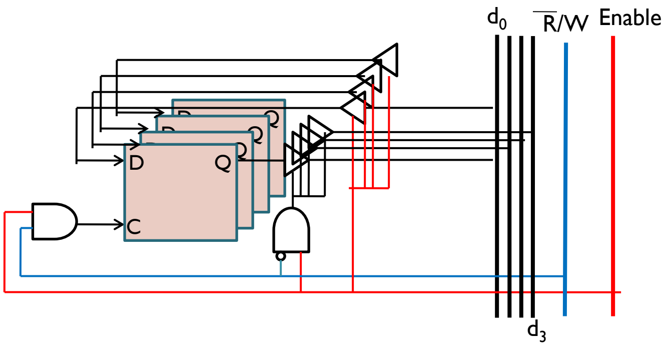
A Static Memory Cell



- *Tri-state Logic:* The output of a gate can be in one of three states – one, zero or not connected

6

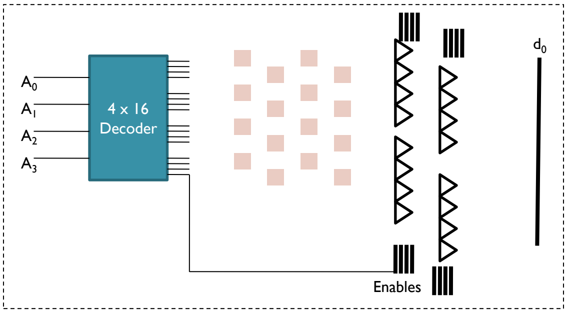
4-bit Register

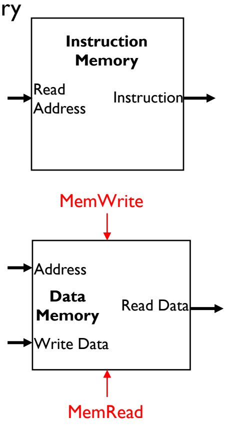


7

What more blocks do we need?

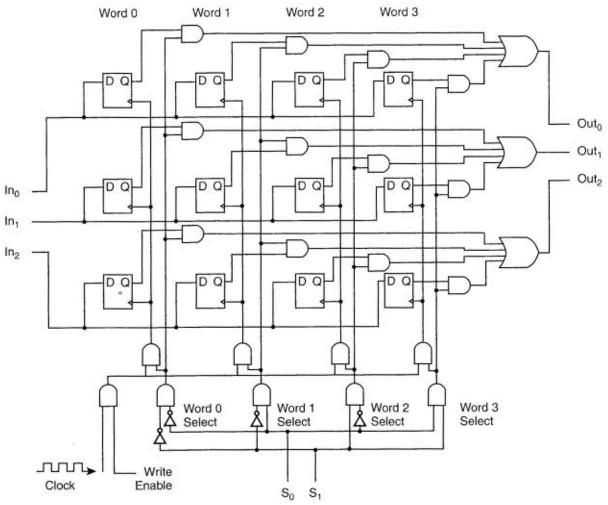
- We need memory to store instructions and data
  - Instruction memory takes address and supplies instructions
  - Data memory takes address and supply data (eg. load)
  - Data memory takes address and data and write into memory





8

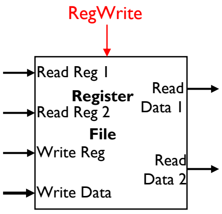
Sample 4 x 3 Memory

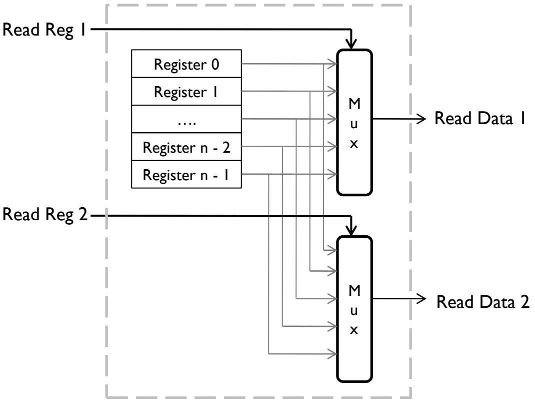


http://www.sci.brooklyn.cuny.edu/~jones/cisc3310/Null%20&%20Lobur%20Figures.htm

Other Blocks?

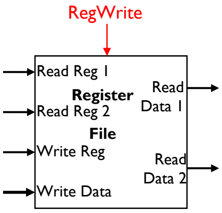
- We need register file to include 32 registers
  - 2 port read for a register file

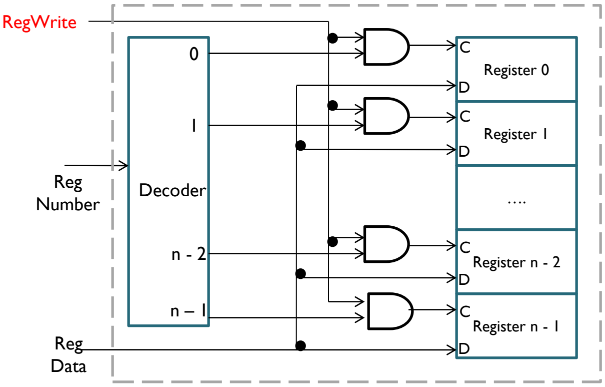




Other Blocks?

- We need register file to include 32 registers
  - Implement write port with write control



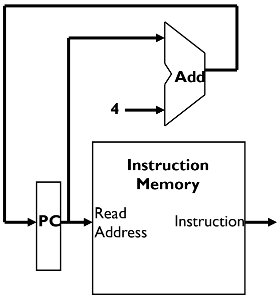


Any other blocks?

- PC (program counter)
- Add support for immediate class of instructions
- Add support for J, JR, JAL

Fetching Instructions

- Fetching instructions involves
  - reading the instruction from the Instruction Memory
  - updating the PC to hold the address of the next instruction

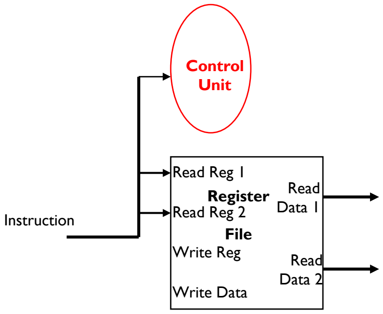


  - PC is updated every cycle, so it does not need an explicit write control signal
  - Instruction memory is read every cycle, so it doesn't need an explicit read control signal

13

Decoding Instructions

- Decoding instructions involves
  - sending the fetched instruction's opcode and function field bits to the control unit

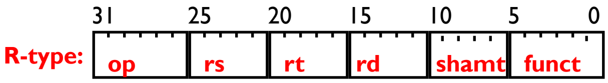


  - reading two values from the Register File
    - Register File addresses are contained in the instruction

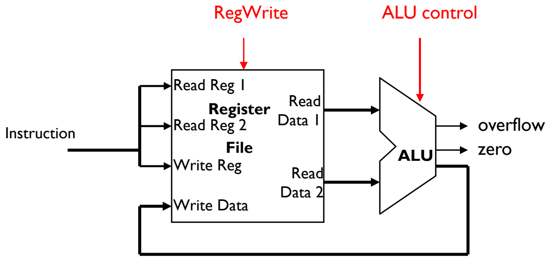
14

Executing R Format Operations

- R format operations (**add, sub, slt, and, or**)



  - perform the (op and funct) operation on values in rs and rt
  - store the result back into the Register File (into location rd)

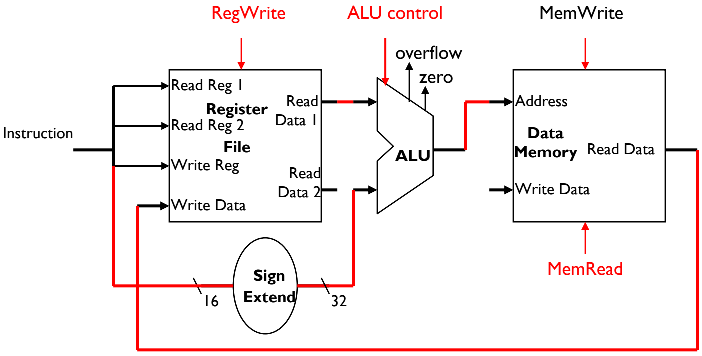


- The Register File is not written every cycle (e.g. **sw**), so we need an explicit write control signal for the Register File

15

Executing Load Operation

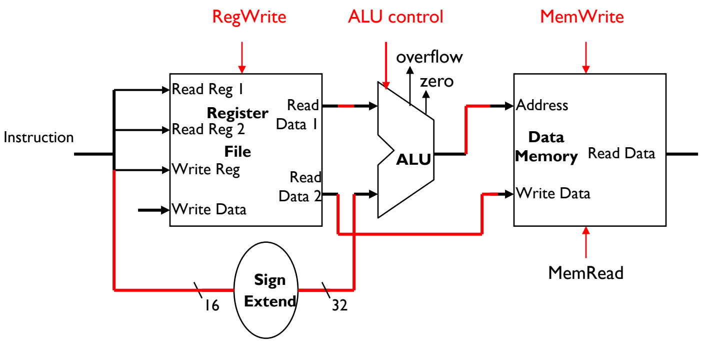
- Load and store operations involves
  - compute memory address by adding the base register (read from the Register File during decode) to the 16-bit signed-extended offset field in the instruction
  - load value, read from the Data Memory, written to the Register File



16

Executing Store Operation

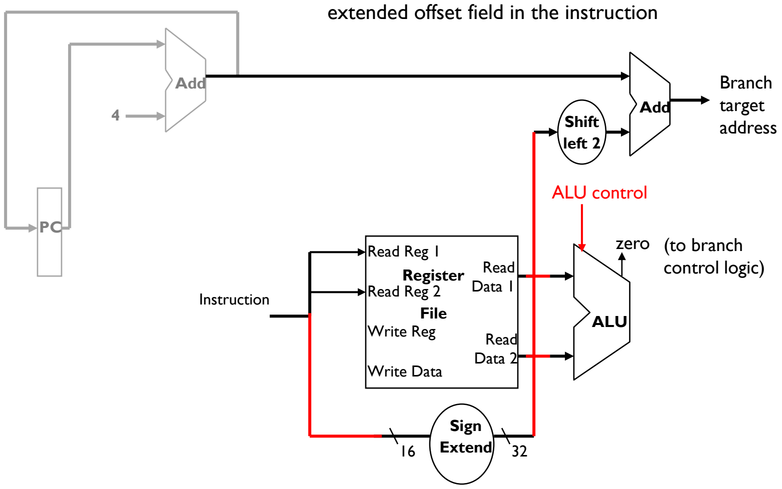
- Load and store operations involves
  - compute memory address by adding the base register (read from the Register File during decode) to the 16-bit signed-extended offset field in the instruction
  - store value (read from the Register File during decode) written to the Data Memory



17

Executing Branch Operations

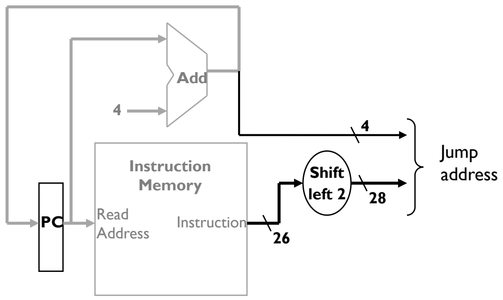
- Branch operations involves
  - compare the operands read from the Register File during decode for equality (`zero` ALU output) compute the branch target address by adding the updated PC to the 16-bit signed-extended offset field in the instruction



18

Executing Jump Operations

- Jump operation involves
  - replace the lower 28 bits of the PC with the lower 26 bits of the fetched instruction shifted left by 2 bits



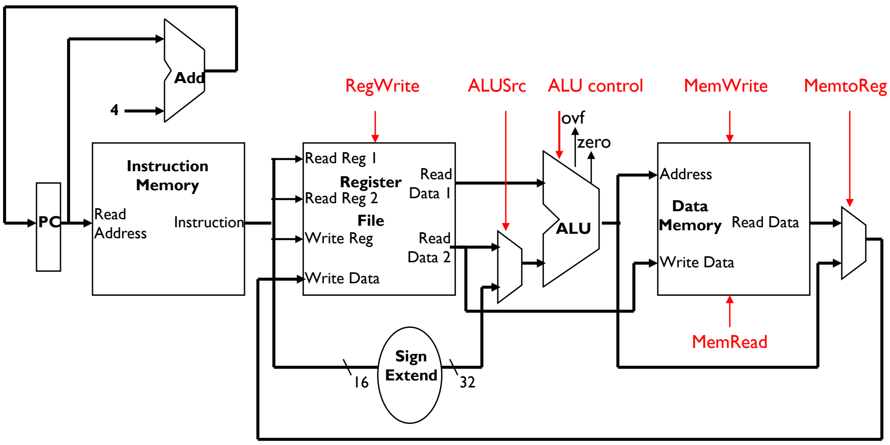
19

Creating a Single Datapath from the Parts

- Assemble the datapath segments and add control lines and multiplexors as needed
- Single cycle design – fetch, decode, and execute each instructions in one cycle
  - no datapath resource can be used more than once per instruction, so some must be duplicated (Eg. separate Instruction Memory and Data Memory, several adders)
  - multiplexors needed at the input of shared elements with the control lines to do the selection
  - write signals to control writing to the Register File and Data Memory
- Cycle time is determined by length of the longest path

20

Fetch, R, and Memory Access Portions



Adding the Control

- Selecting the operations to perform (ALU, Register File, Memory read/write)
  - Select the registers to be read (always two)
  - Select the 2nd ALU input
  - Select the operation to be performed
  - Select the data memory to be written

- Controlling the flow of data (multiplexor inputs)

Adding the Control

	31	25	20	15	10	5	0
R-type:	**op**	**rs**	**rt**	**rd**	**shamt**	**funct**	

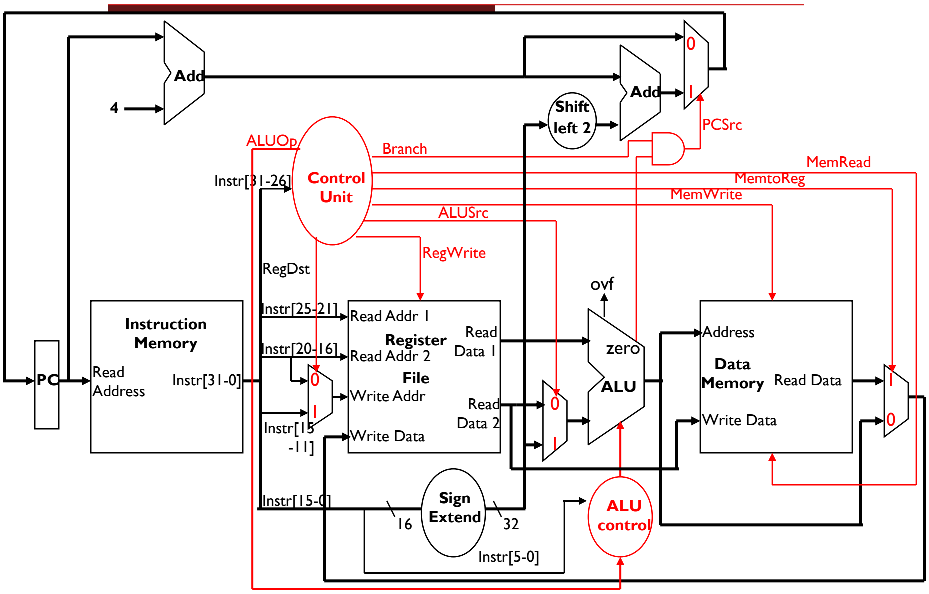
	31	25	20	15	0
I-Type:	**op**	**rs**	**rt**	**address offset**	

	31	25	0
J-type:	**op**	**target address**	

- Observations
  - op field always in bits 31-26
  - address of registers to be read are always specified by the rs field (bits 25-21) and rt field (bits 20-16); for lw and sw, rs is the base register
  - addr. of register to be written is in one of two places – in rt (bits 20-16) for lw; in rd (bits 15-11) for R-type instructions
  - offset for beq, lw, and sw always in bits 15-0

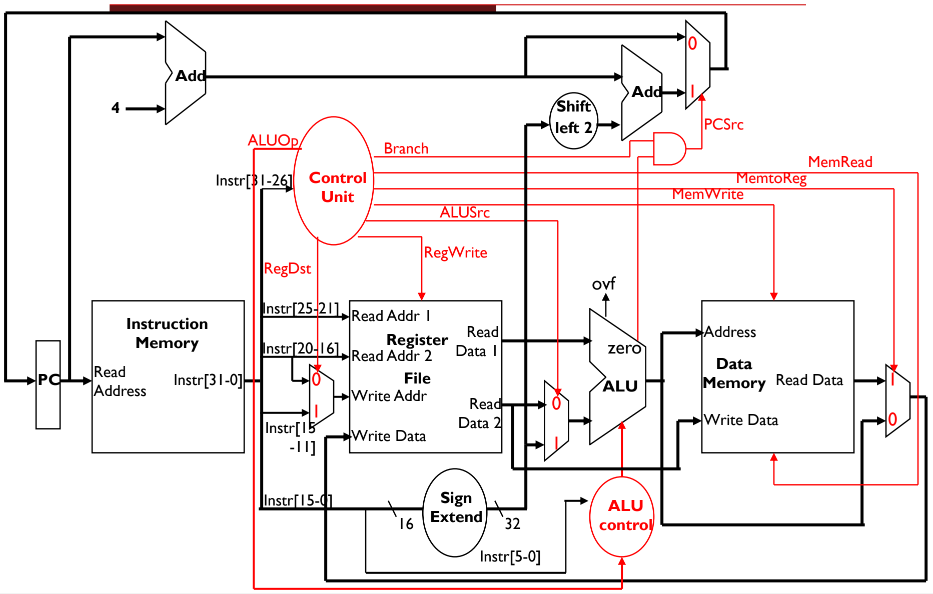
23

Single Cycle Datapath with Control Unit



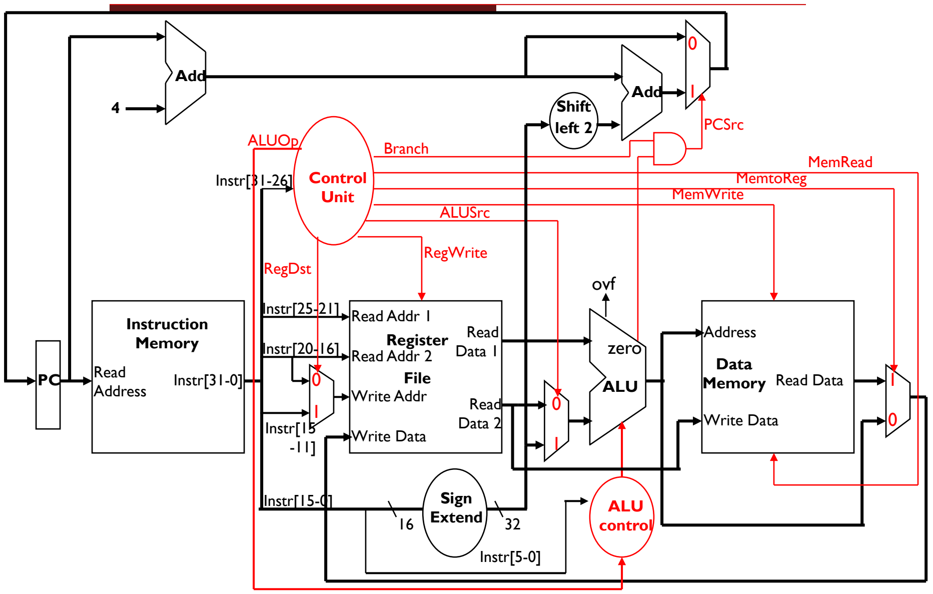
24

Load Word Instruction Data/Control Flow



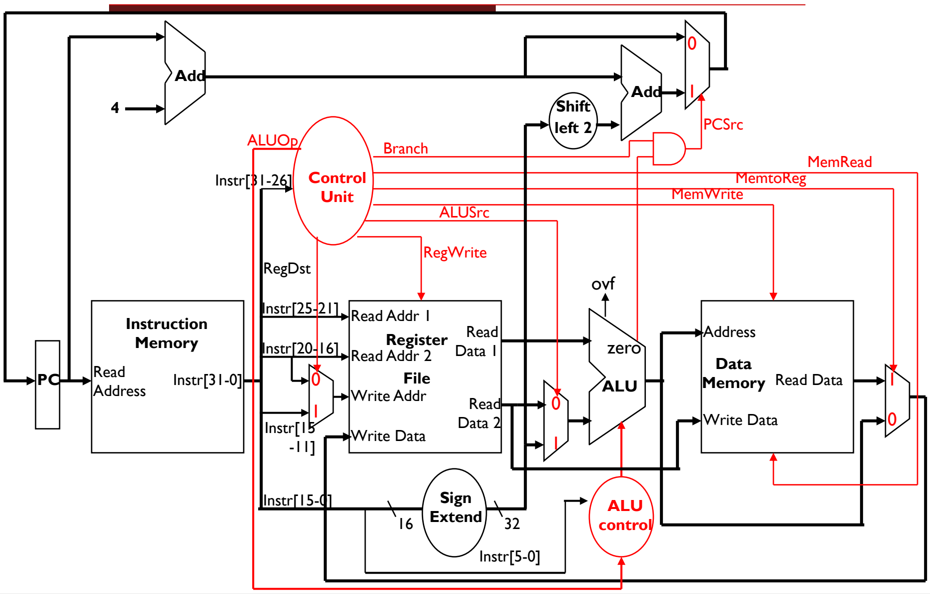
25

Branch Instruction Data/Control Flow



26

Adding the Jump Operation



27