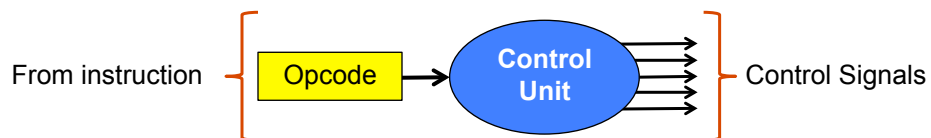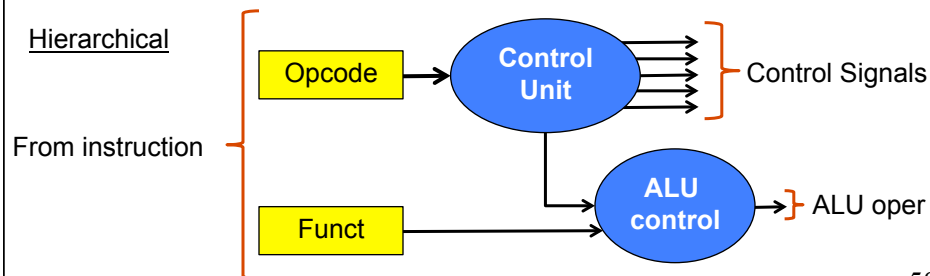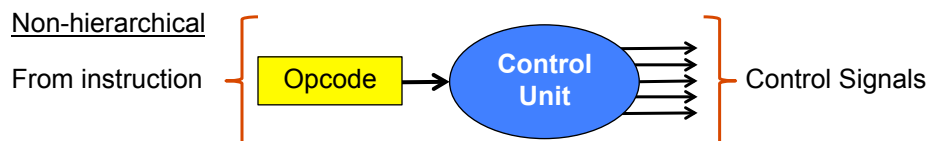## What about all those "control" signals?

- **Need to set control signals, e.g., muxes, register write, memory operations, etc.**

- **Control Unit: Combinational logic that "decodes" instruction opcode to determine control signals**

From instruction — [Opcode] → (Control Unit) ⇒ Control Signals

57

Hierarchical Control Unit

- **MIPS uses multiple control units**
- **Units are hierarchical**

Non-hierarchical

From instruction — [Opcode] → (Control Unit) ⇒ Control Signals

Hierarchical

[Opcode] → (Control Unit) ⇒ Control Signals

From instruction

[Funct] → (ALU control) → ALU oper

58

The Control Unit

- **Decodes instruction to determine what segments will be active in the datapath**

- **Generates signals to**
  - **Set muxes to correct input**
  - **Operation code to ALU**
  - **Read and write to register file**
  - **Read and write to memory (load/store)**
  - **Update of program counter (branches)**
  - **Branch target address computation**

- **Two parts: ALU control and Main control (muxes, etc)**

59

ALU Control

- **ALU control: specifies what operation ALU performs**
  - **I.e., *ALU operation* control signals**
  - **Eight input combinations (3 input control signals)**
  - **Five combinations used to select operation**
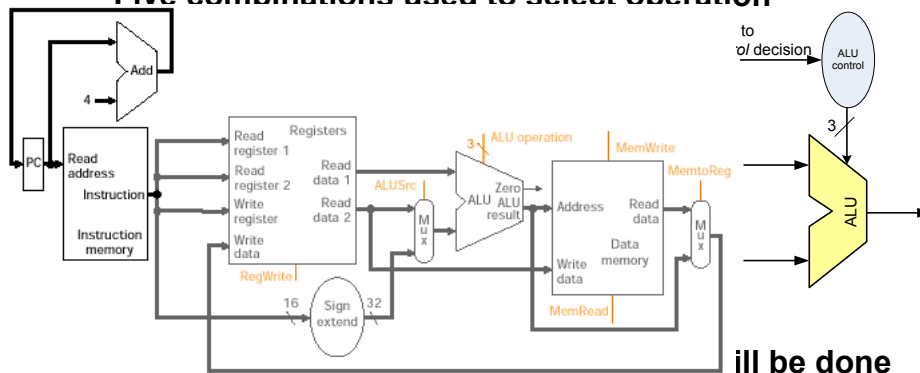
ill be done
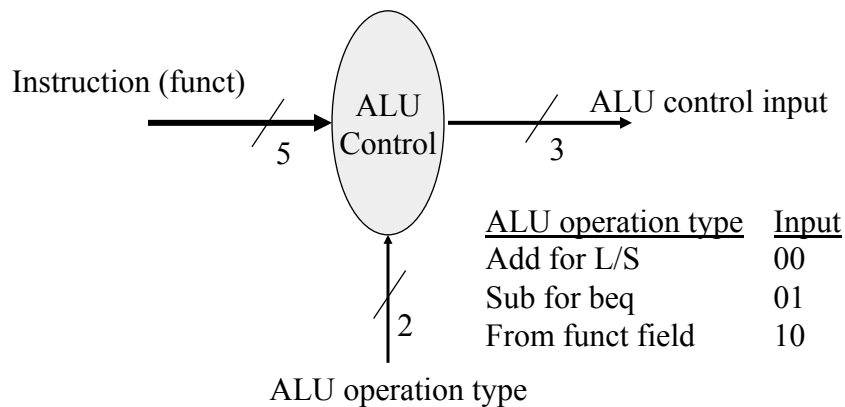
60

ALU Control - Selecting Operation

Class (Opcode)	Operation		Control
Load/store	Addition (memory address)		010
Branch	Subtraction (comparison)		110
Arithmetic	Depends on funct field:		
	100000	add	010
	100010	subtract	110
	100100	and	000
	100101	or	001
	101010	set on less than	111

- **Generate ALU control based on opcode and funct field**

ALU Control Unit

- **Small control unit associated with ALU**
  - **Generates appropriate control signals to ALU**

Instruction (funct)

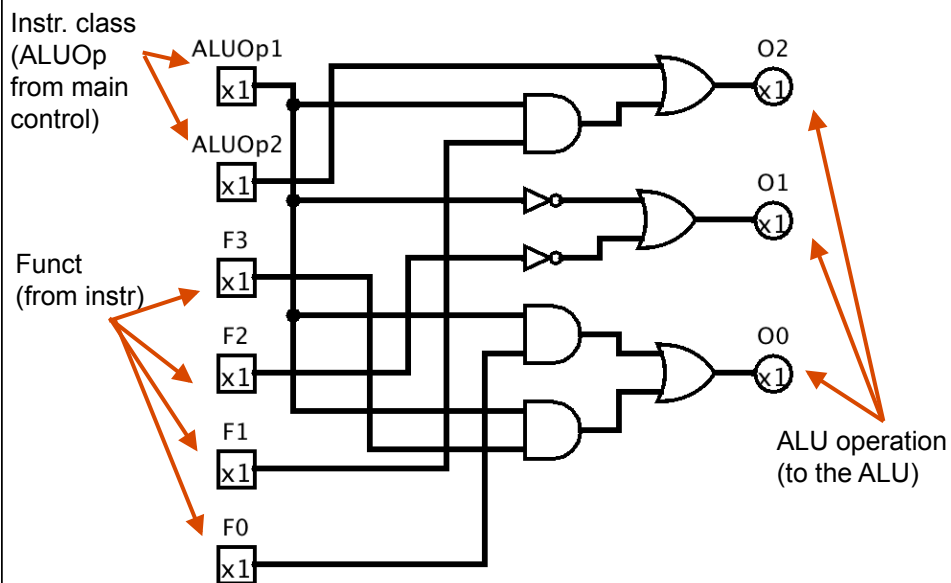ALU Control

5

ALU control input

3

2

ALU operation type

ALU operation type	Input
Add for L/S	00
Sub for beq	01
From funct field	10

Building the ALU Control Unit

- **Use truth table to determine how output will be generated based on the inputs**

Operation	ALUOp	Funct	Output
Load/Store	00	XXXXXX	010 (add)
Beq	X1	XXXXXX	110 (sub)
Arithmetic	1X	XX0000	010 (add)
	1X	XX0010	110 (sub)
	1X	XX0100	000 (and)
	1X	XX0101	001 (or)
	1X	XX1010	111 (slt)

- **From truth table, we can derive the control circuit**

63

ALU Control – Created in Logisim

Instr. class (ALUOp from main control)

ALUOp1

ALUOp2

Funct (from instr)

F3

F2

F1

F0

O2

O1

O0

ALU operation (to the ALU)

66

Main Control Unit

- **Use fields from instruction to generate control**
  - **We will "connect" the fields of the instruction to the datapath via the main control unit**

R-type instruction

0	rs	rt	rd	shamt	funct
31-26	25-21	20-16	15-11	10-6	5-0

Load/Store

35 / 43	rs	rt	address
31-26	25-21	20-16	15-0

Branch (beq)

4	rs	rt	address
31-26	25-21	20-16	15-0

67

Main Control Unit

- **Use fields from instruction to generate control**
  - **We will "connect" the fields of the instruction to the datapath via the main control unit**

R-type instruction

0	rs	rt	rd	shamt	funct
31-26	25-21	20-16	15-11	10-6	5-0

Load/Store

35 / 43	rs	rt	address
31-26	25-21	20-16	15-0

Branch (beq)

4	rs	rt	address
31-26	25-21	20-16	15-0

Opcode is always in same position (31-26), called "Op[5-0]"

68

Main Control Unit

- **Use fields from instruction to generate control**
  - **We will "connect" the fields of the instruction to the datapath via the main control unit**

R-type instruction

0	rs	rt	rd	shamt	funct
31-26	25-21	20-16	15-11	10-6	5-0

Load/Store

35 / 43	rs	rt	address
31-26	25-21	20-16	15-0

Branch (beq)

4	rs	rt	address
31-26	25-21	20-16	15-0

Registers to be read are always rs and rt (always in fixed place)

69

Main Control Unit

- **Use fields from instruction to generate control**
  - **We will "connect" the fields of the instruction to the datapath via the main control unit**

R-type instruction

0	rs	rt	rd	shamt	funct
31-26	25-21	20-16	15-11	10-6	5-0

Load/Store

35 / 43	rs	rt	address
31-26	25-21	20-16	15-0

Branch (beq)

4	rs	rt	address
31-26	25-21	20-16	15-0

Base register for load/store is always rs in position 25-21

70

Main Control Unit

- **Use fields from instruction to generate control**
  - **We will "connect" the fields of the instruction to the datapath via the main control unit**

R-type instruction

0	rs	rt	rd	shamt	funct
31-26	25-21	20-16	15-11	10-6	5-0

Load/Store

35 / 43	rs	rt	address
31-26	25-21	20-16	15-0

Branch (beq)

4	rs	rt	address
31-26	25-21	20-16	15-0

16-bit offset for branch equal, load, and store always in 15-0

71

Main Control Unit

- **Use fields from instruction to generate control**
  - **We will "connect" the fields of the instruction to the datapath via the main control unit**

R-type instruction

0	rs	rt	rd	shamt	funct
31-26	25-21	20-16	15-11	10-6	5-0

Load/Store

35 / 43	rs	rt	address
31-26	25-21	20-16	15-0

Branch (beq)

4	rs	rt	address
31-26	25-21	20-16	15-0

Destination register in one of two places: 15-11 for arithmetic and 20-16 for load; need multiplexor on write register address

72

Full Datapath and Control Signals

- **Control includes four muxes, ALU control unit, and control to register file and data memory**


The Control Signals

- **Two ALUOp signals**

- **Seven other signals**
  - **RegDst** - which field for write register
  - **RegWrite** - write to register file
  - **ALUSrc** - source for second ALU input
  - **PCSrc** - source for PC (PC + 4 or target address)
  - **MemRead** - read input address from memory
  - **MemWrite** - write input address/data to memory
  - **MemToReg** - source of write register port data input

- **Branch control signal (set when instruction is branch)**

Full Datapath and Full Control

75
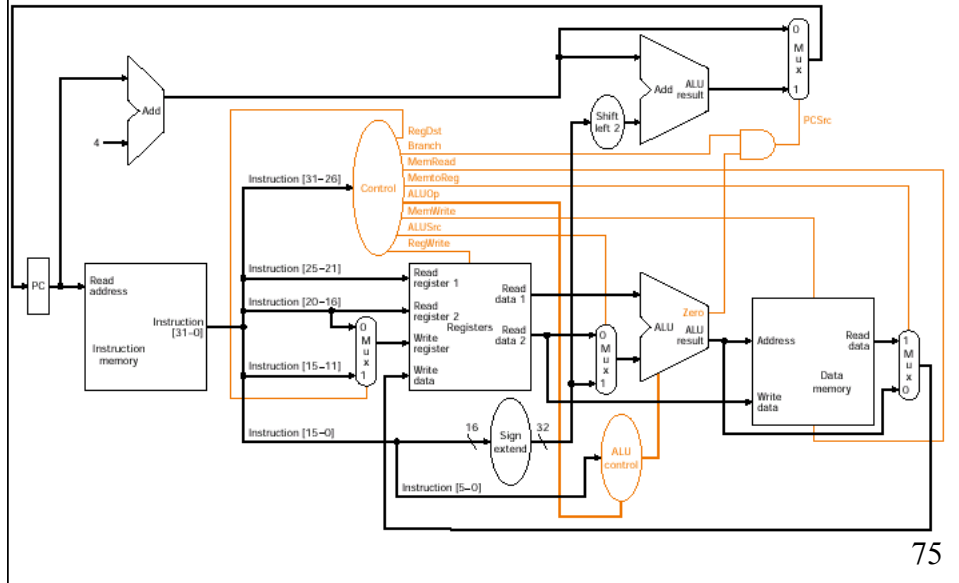
Operation - R-type instructions
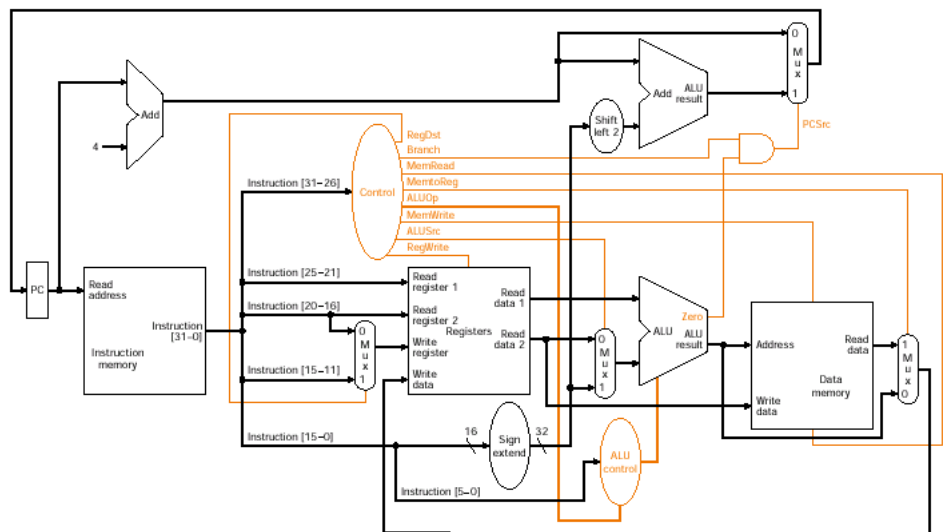
Let's do a add $t1,$t2,$t3

76
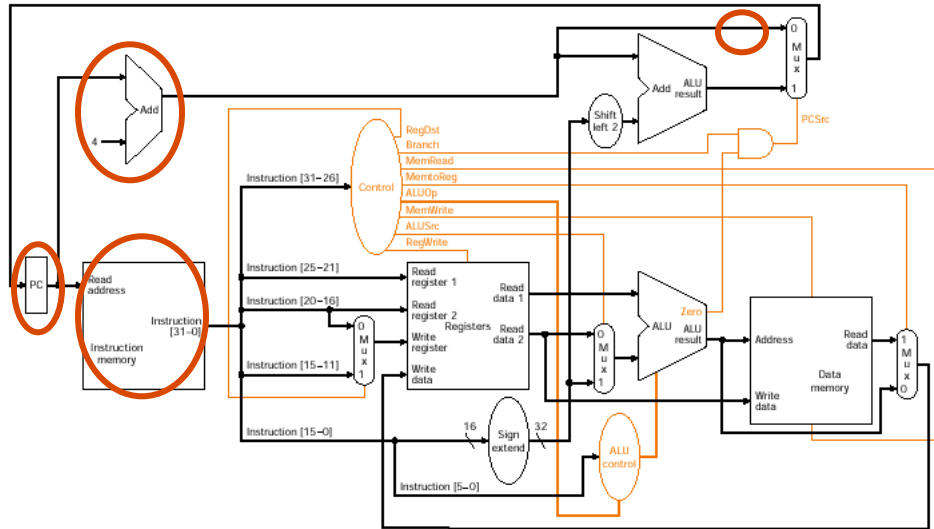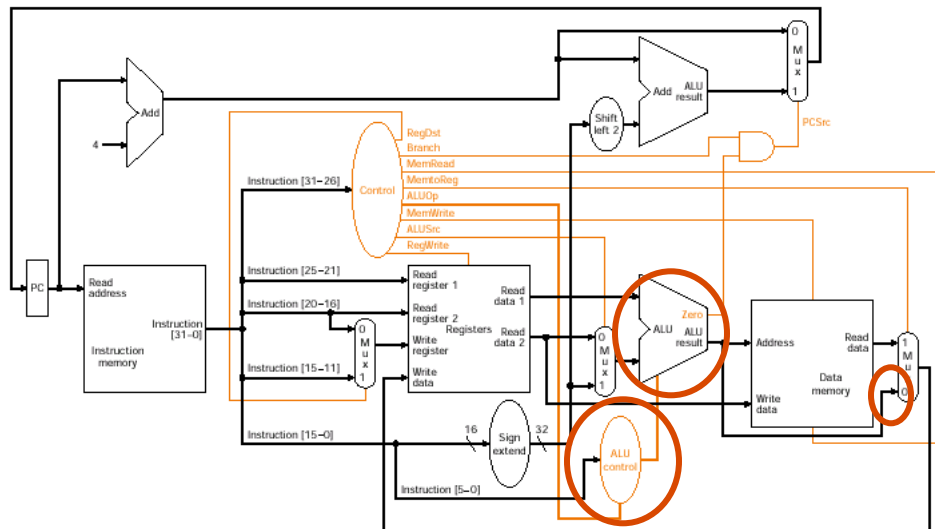
FETCH - add $t1,$t2,$t3

On fetch - PC, increment PC by 4, read from instruction memory   77

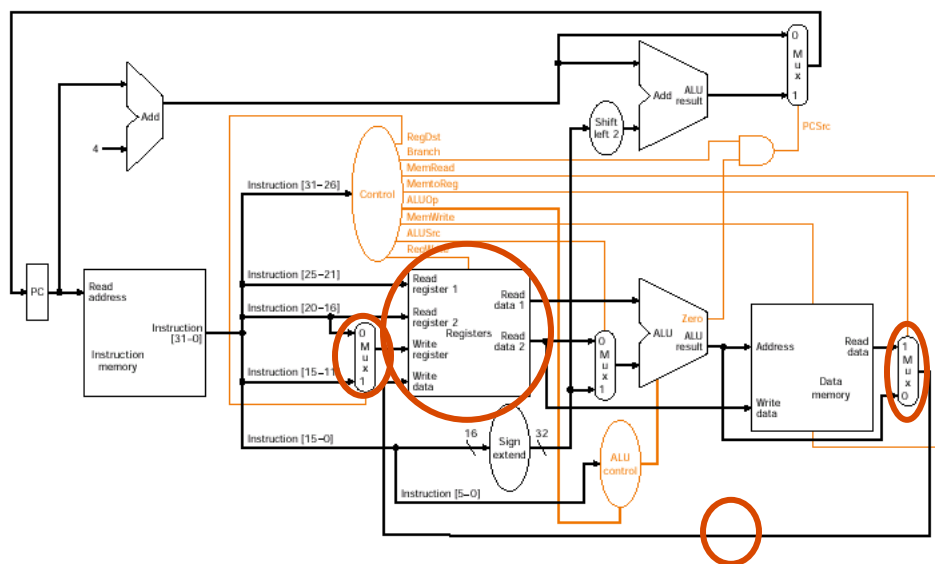# READ REGISTERS (t1,t2) - add $t1,$t2,$t3
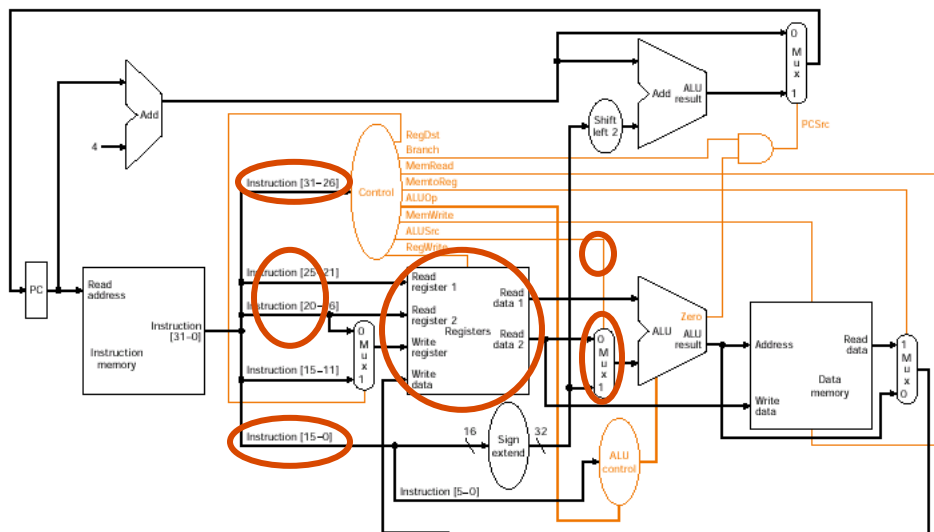
Read source registers, main control unit computed all settings   78

EXECUTE - add $t1,$t2,$t3

ALU operates on data from register file, ALU control determined   79

WRITE DESTINATION (t3) - add $t1,$t2,$t3


Result from ALU is written to register file   80

Remember...Combinational Single Cycle

- **Distinct steps shown for clarity**
- **Reality - information flows in those steps but it's all combinational logic**
- **Signals within the datapath vary and stabilize roughly in the flow of steps given**

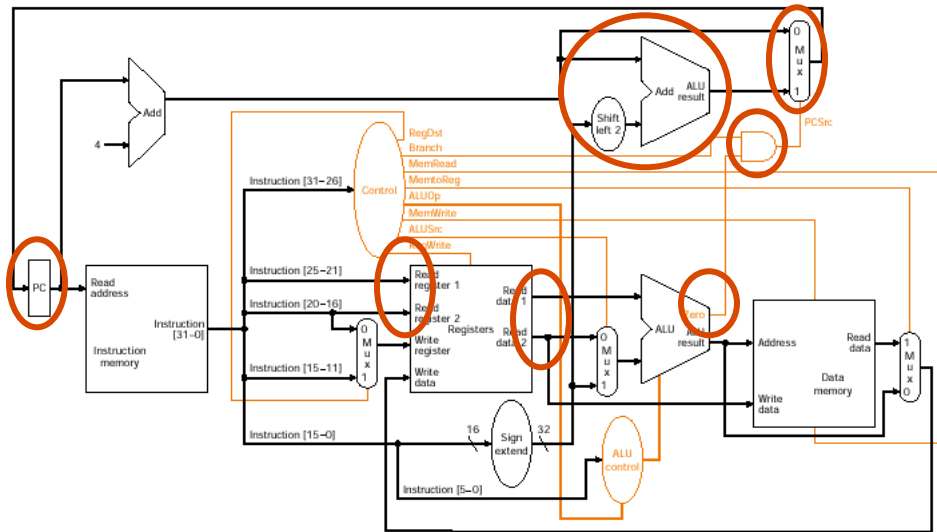- **All units and paths as marked during each step are active throughout the process!**

Load Word - lw $t1, offset($t2)
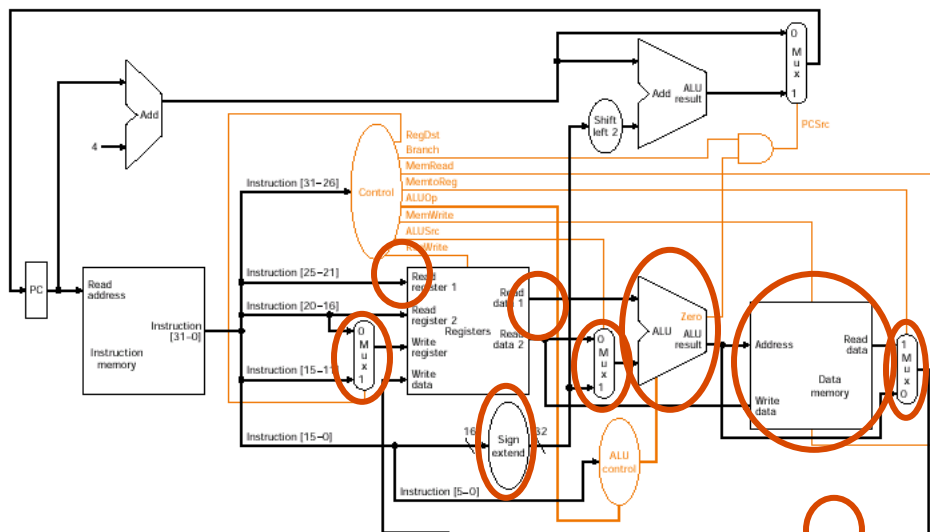
Only one source reg to read, sign extend, form address, write to destination in different position in instruction (bits 20-16)
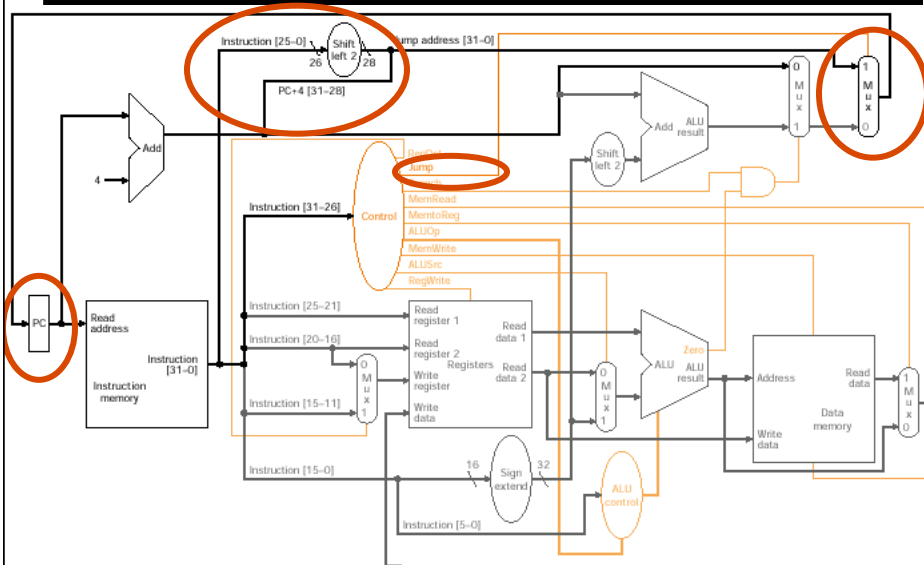
Branches - beq $t1,$t2,offset

Read and compare sources, generate zero & branch control, compute target address, update PC if zero and branch

83

Jumps - j label

Shift address, concatenate with PC, extra mux (three possible sources now for program counter)

84