CSC258: Computer Organization

Microarchitecture

Review: Quiz-like Questions

1. Design a greater-than-or-equal-to comparator for 32-bit numbers. You may use any combinational building block (adders, logic gates, muxes or demuxes) that you wish *except for another comparator*. Your circuit should output a single-bit value, *gte*, which has the value 1 iff 32-bit input A is greater than or equal to 32-bit input B. Sketch the schematic for your comparator.

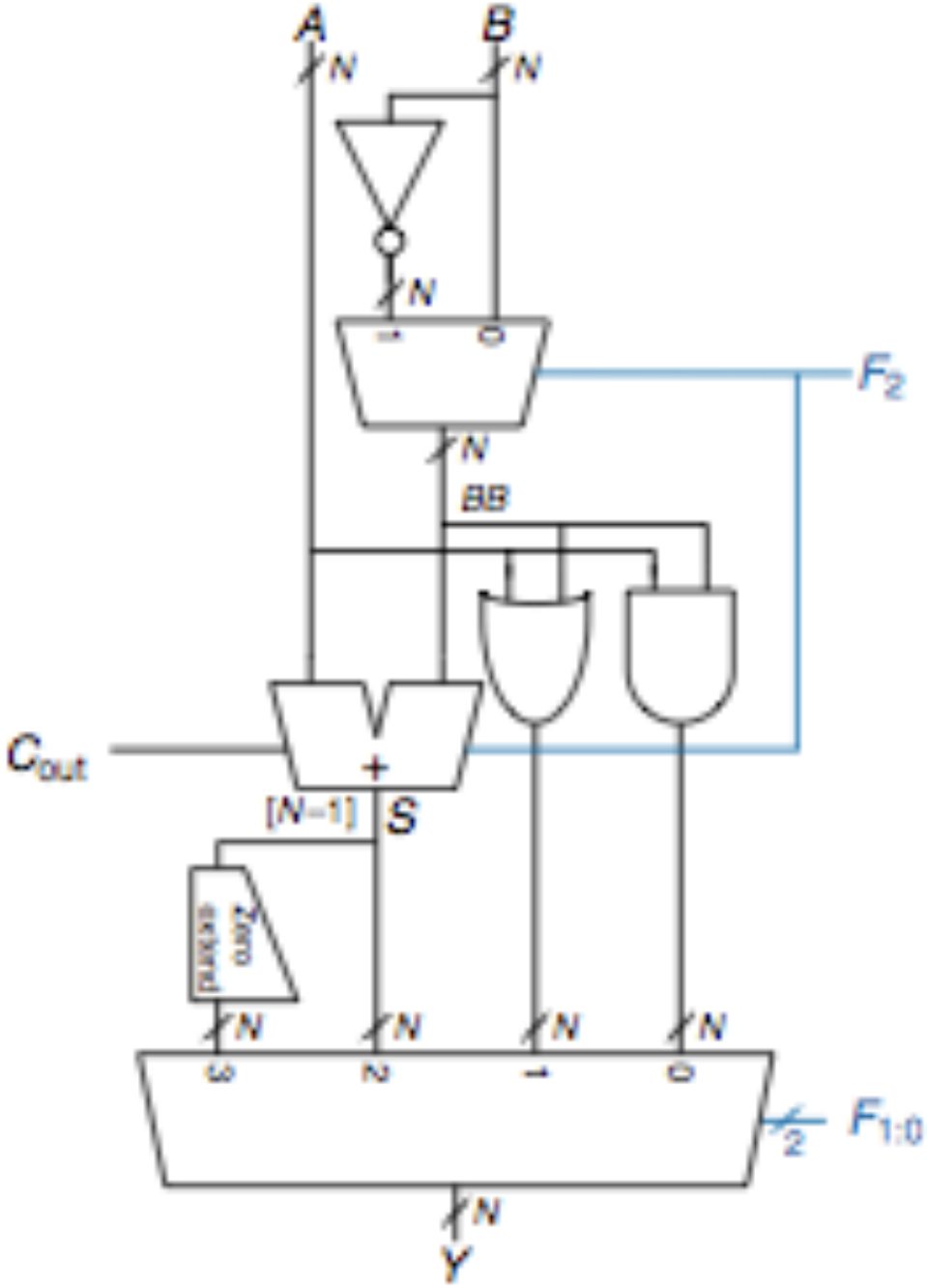
2. In one sentence, succinctly define the role of the *datapath* of a processor.

Microarchitecture

Key Questions

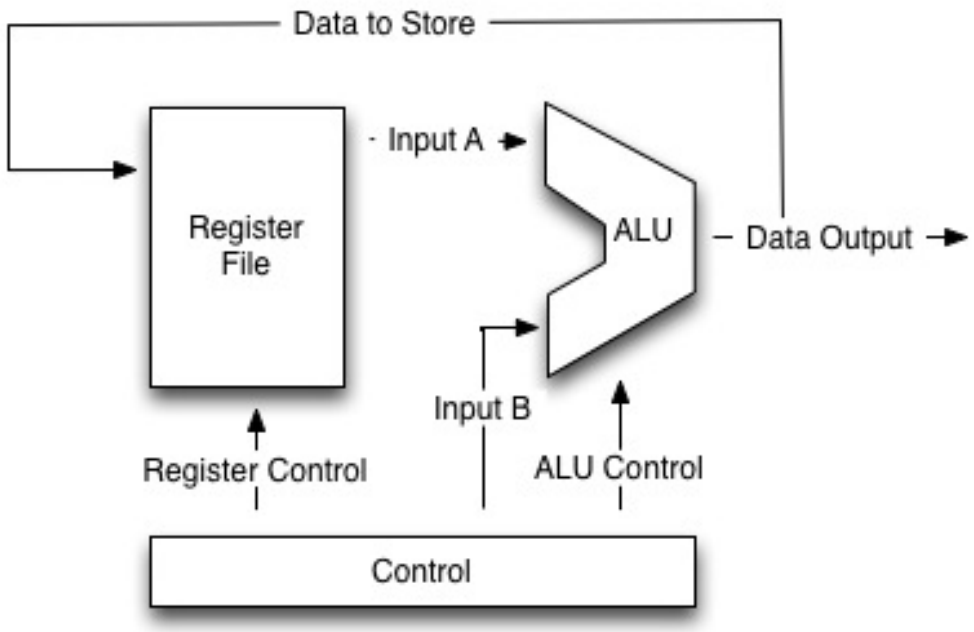
- Do I understand the datapath of the ALU in figure 5.15?
- What is the difference between data and control?
- How do I compute the performance of a processor?
- Do I understand the development of the single-cycle MIPS datapath (Figure 7.11)?

ALU Schematic: Figure 5.15 from DDCA



The Processor from Lab

- This circuit is a FSM.
- The state is the register file.
- The ALU provides flexible computation.



- What is control?
- And where does the data go?

Control: Instruction Fetch and Decode

- The instruction decoder is fed instructions one at a time.
  - It outputs the control necessary to make the processor execute the user's instruction.
  - In essence, it's a very fancy decoder.
    - It could even be a FSM in its own right: some instructions are *microcoded* to be implemented by more than one smaller instruction.
- But where do the instructions come from and what do they look like?

Control: Instruction Fetch and Decode

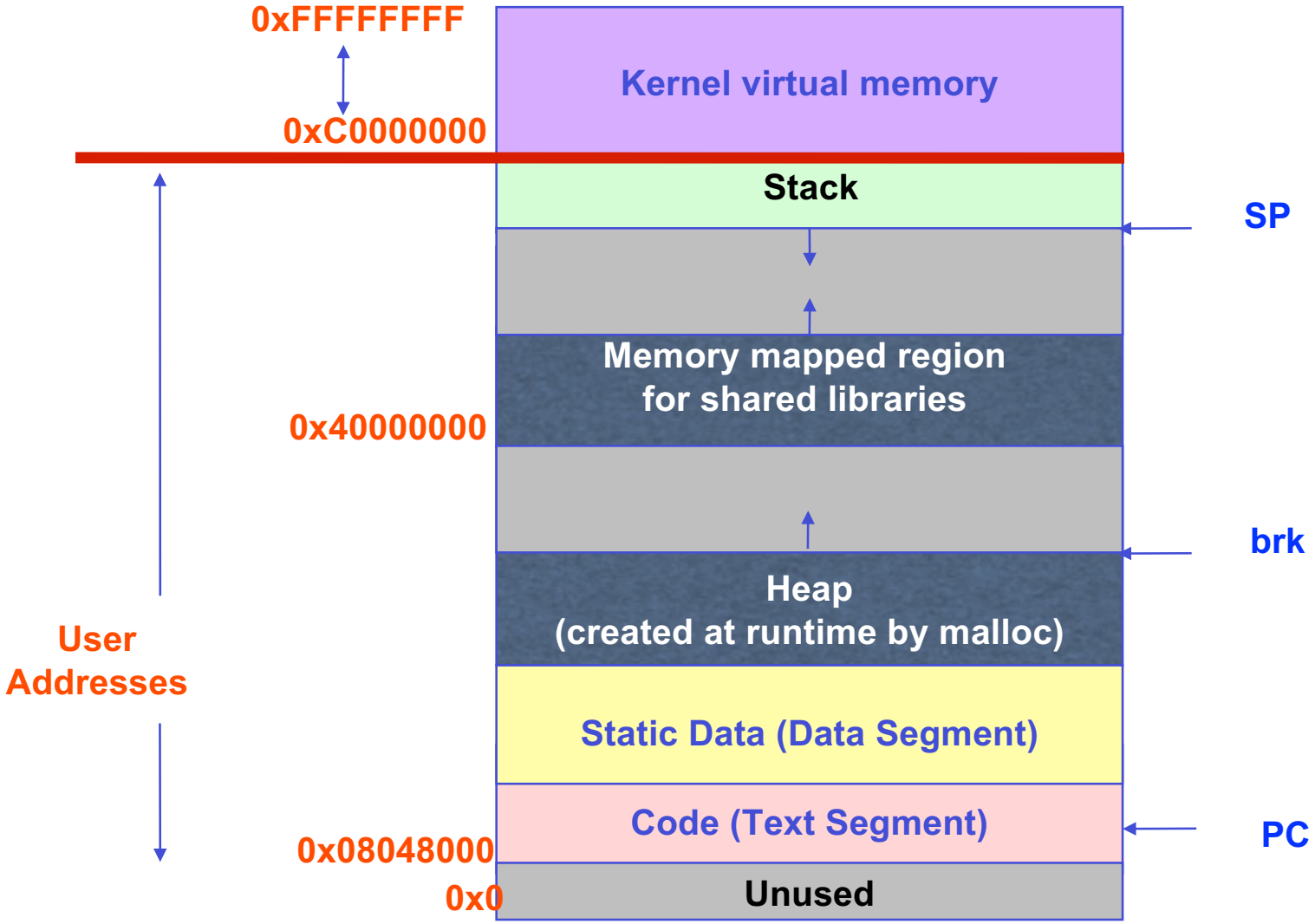
- The instruction decoder is fed instructions one at a time.
  - It outputs the control necessary to make the processor execute the user's instruction.
  - In essence, it's a very fancy decoder.
    - It could even be a FSM in its own right: some instructions are *microcoded* to be implemented by more than one smaller instruction.
- But where do the instructions come from and what do they look like?

Instructions? From where?

- Instructions are encoded in binary. The *compiler tool chain* translates a user's program into machine code.

- The program (in machine code form) is stored in *memory*.
- Memory is a very large storage device. It contains the program as well as data created as the program runs.

Memory Layout (Linux, x86)



The Stack and Heap

- Memory contains two structures that the program can use for extra storage: the stack and heap.
- The pointer to the top of the stack is contained in a hardware register: the stack pointer (SP)
- It is the program's responsibility to manage the stack.
  - *Whenever stack space is allocated, it should be deallocated later by the same function*
- The operating system manages heap allocation.
  - Whenever you *malloc*, you're getting heap space.

Instructions and Data

- Everything in memory is stored in binary.
- There is no good way to determine if an arbitrary binary word is data ... or an instruction.
  - Many hacks use this to their advantage.
- This is actually a feature, not a bug. It means that we can treat instructions and data in the same way.
  - They're *both* data ... we just use instructions to run the machine.

Single-Cycle MIPS *(Credit: DDCA)*

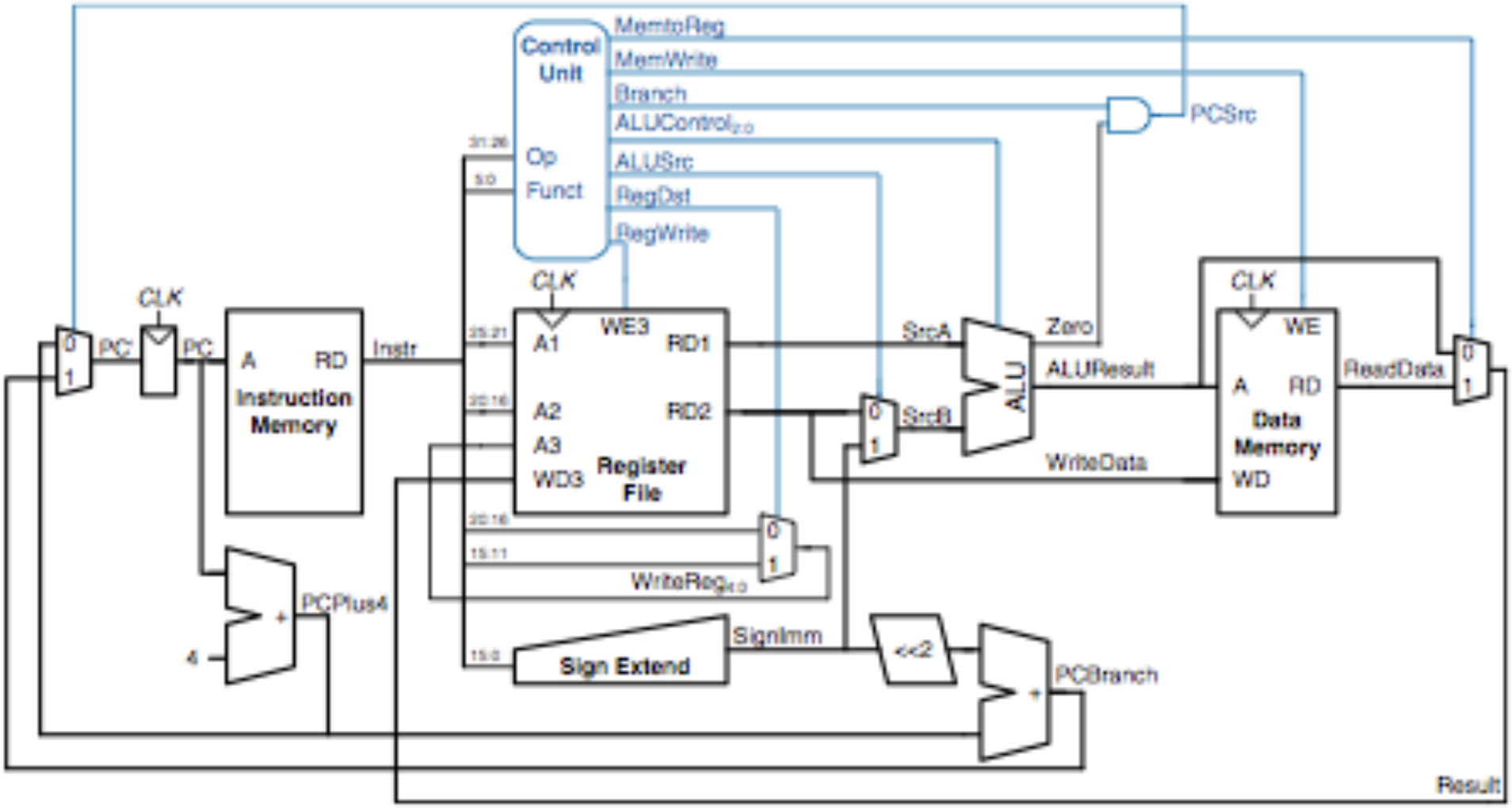


Figure 7.11 **Complete single-cycle MIPS processor**

Key Questions

- Do I understand the datapath of the ALU in figure 5.15?
- What is the difference between data and control?
- How do I compute the performance of a processor?
- Do I understand the development of the single-cycle MIPS datapath (Figure 7.11)?

Executing an "Immediate" Instruction *(Credit: DDCA)*

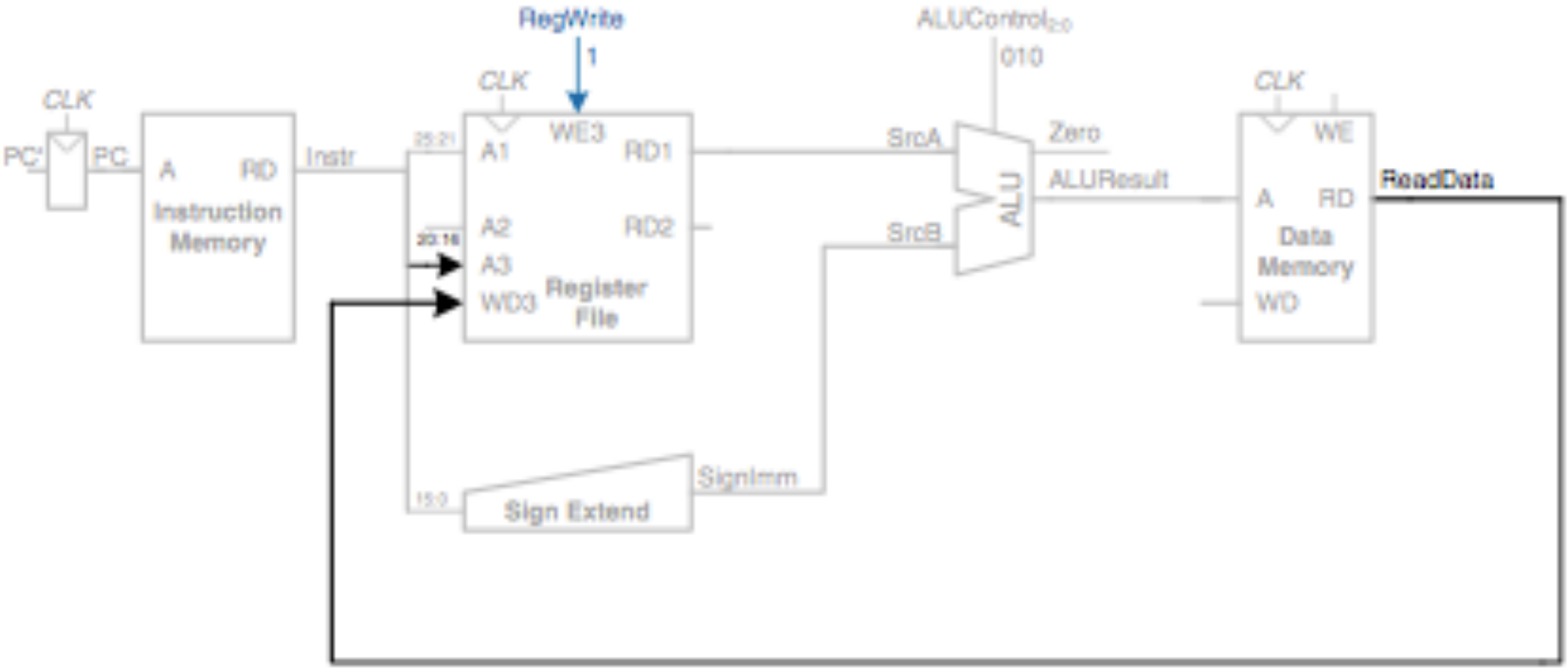


Figure 7.6 Write data back to register file

Executing with Two Registers as Inputs *(Credit: DDCA)*

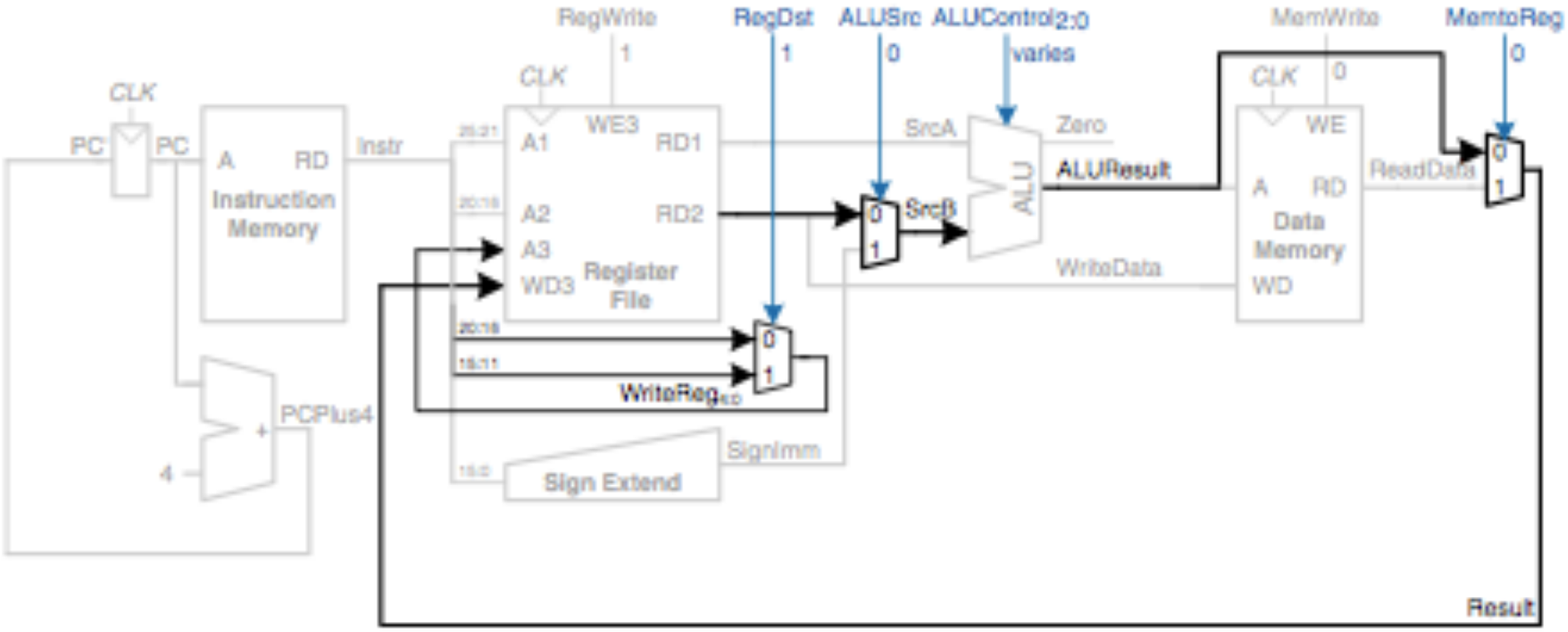


Figure 7.9 Datapath enhancements for R-type instruction

Determining the Next Instruction *(Credit: DDCA)*

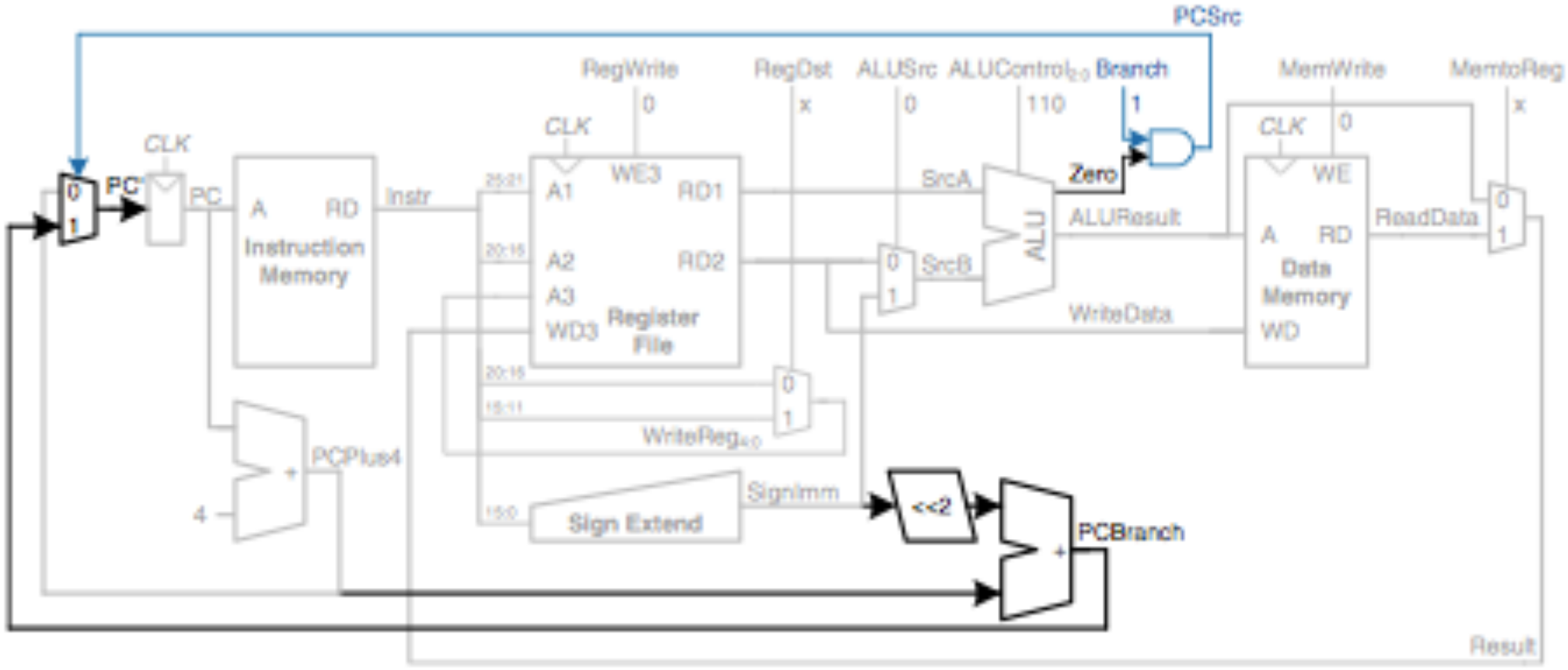


Figure 7.10 Datapath enhancements for `beq` instruction

Executing a Program

- First, load the program into memory.
- Set the program counter (PC) to the first instruction in memory and set the SP to the first empty space on the stack
- Let instruction fetch/decode do the work! The processor can control what instruction is executed next.
- When the process needs support from the operating system (OS), it will "trap" ("throw an exception")
  - Traps include "print" and "halt"

Single-Cycle MIPS *(Credit: DDCA)*

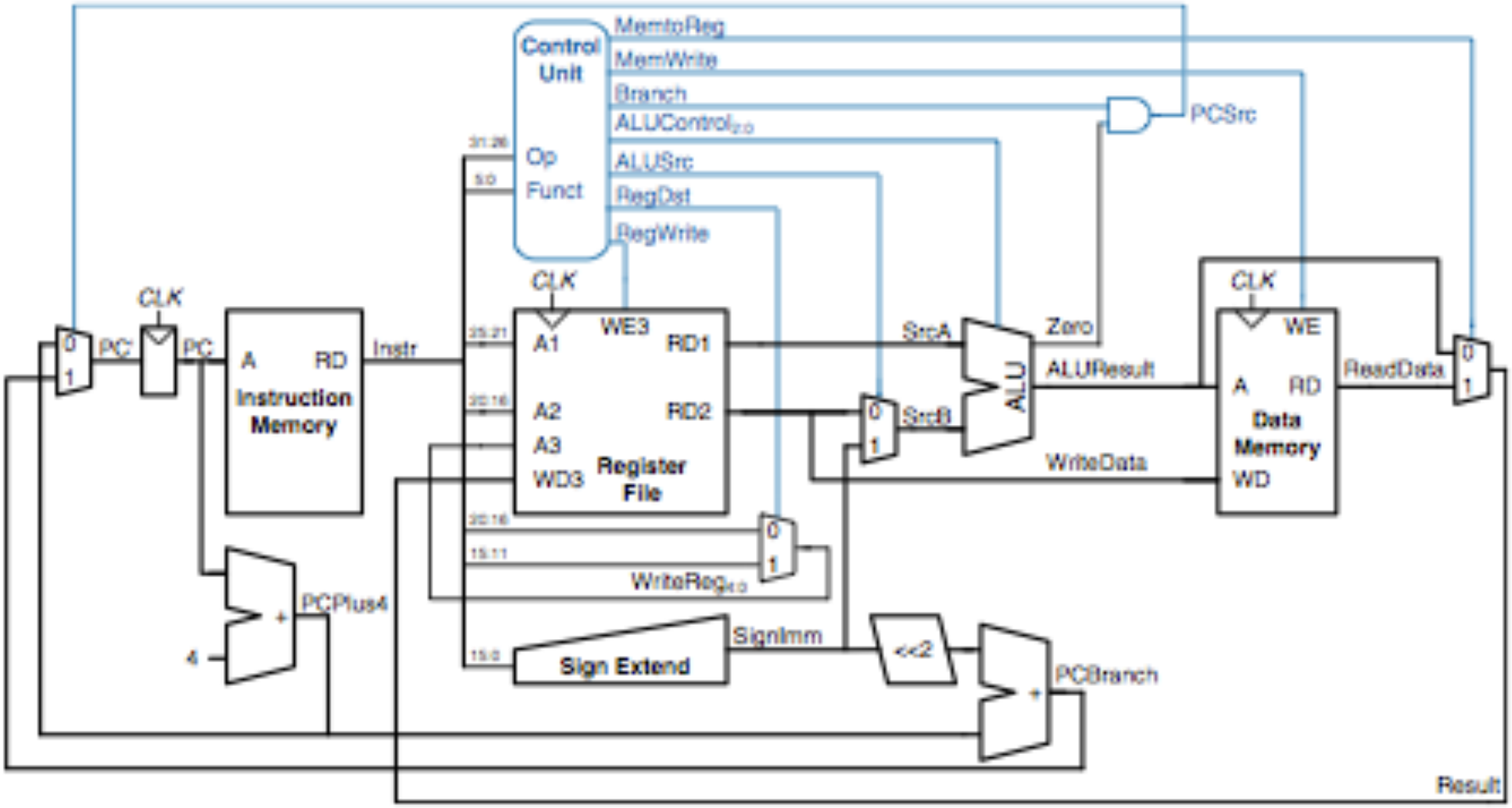


Figure 7.11 **Complete single-cycle MIPS processor**

Check: Quiz-like Questions

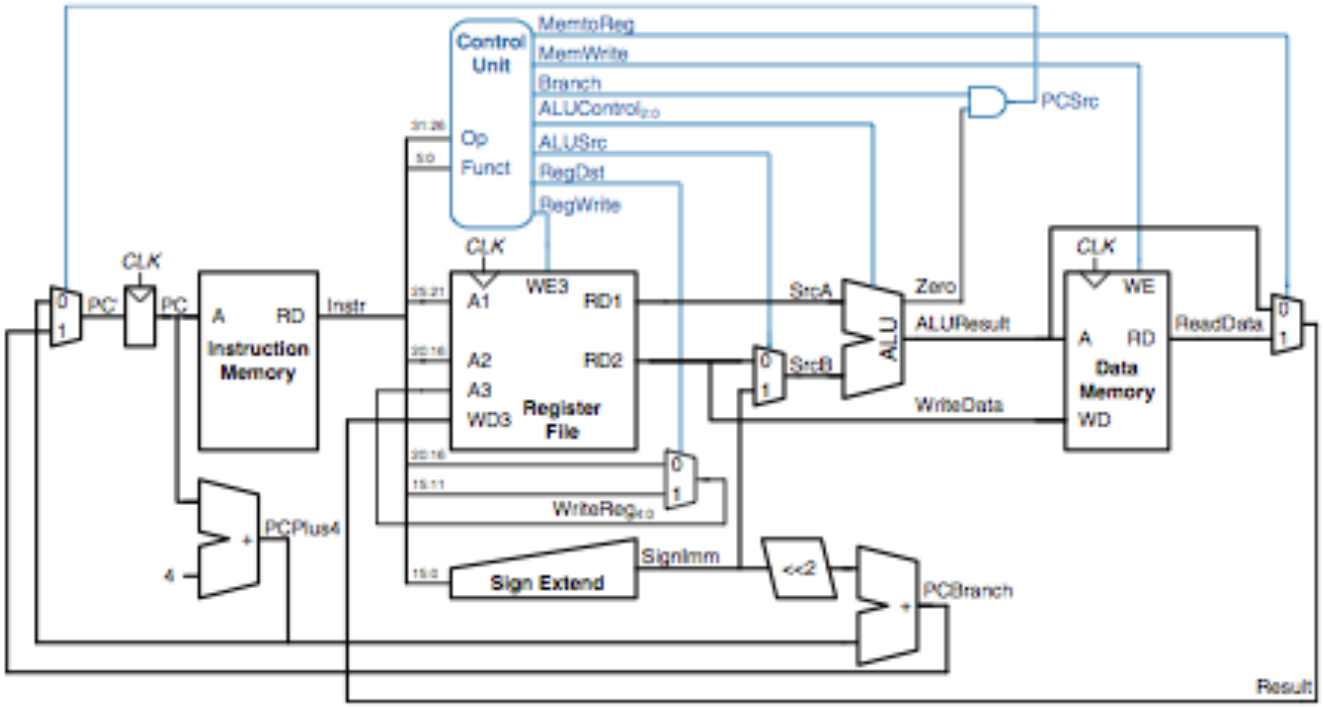


Figure 7.11 **Complete single-cycle MIPS processor**

1. ALUSrc has a stuck at 0 fault. Which instructions will malfunction?
2. RegDst has a stuck at 1 fault. Which instructions will malfunction?

Check: Quiz-like Questions

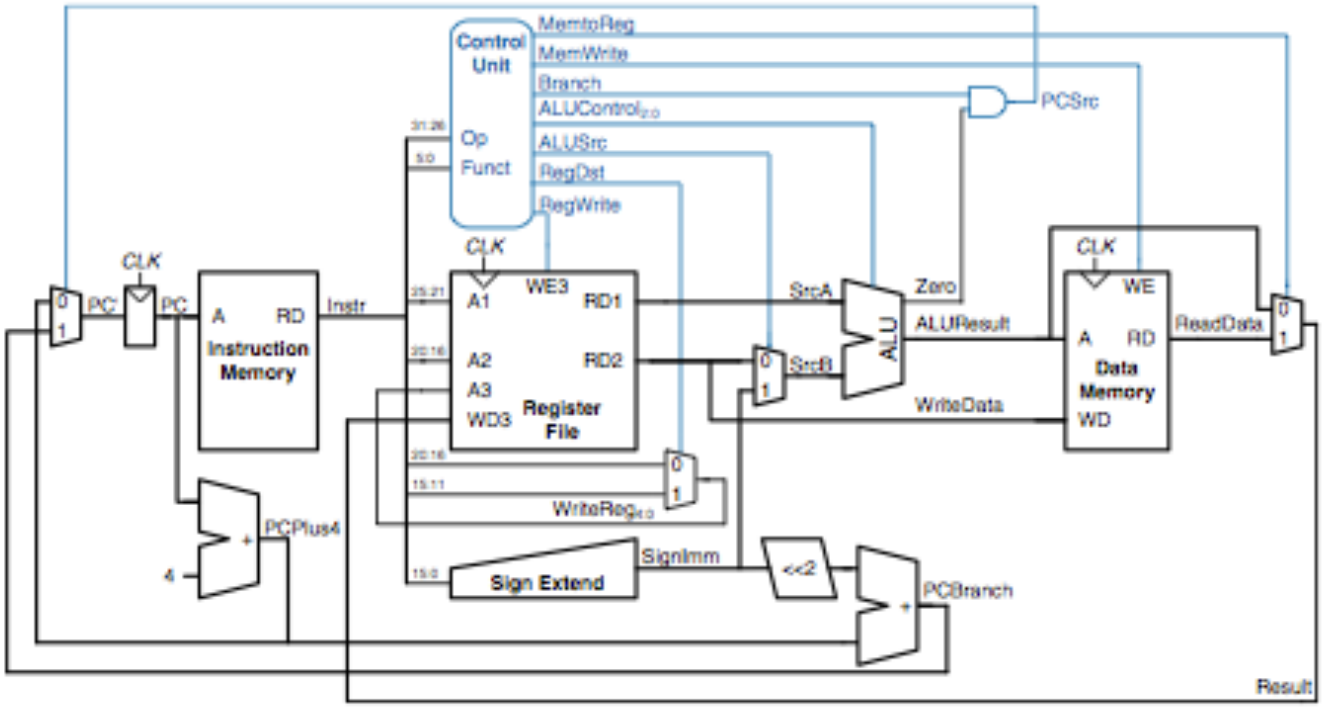


Figure 7.11 **Complete single-cycle MIPS processor**

Provide the values of the control lines required to execute the I-type instruction "subi r12, r10, 1". Assume that the ALU Control for subi is 100.

Check: Quiz-like Questions

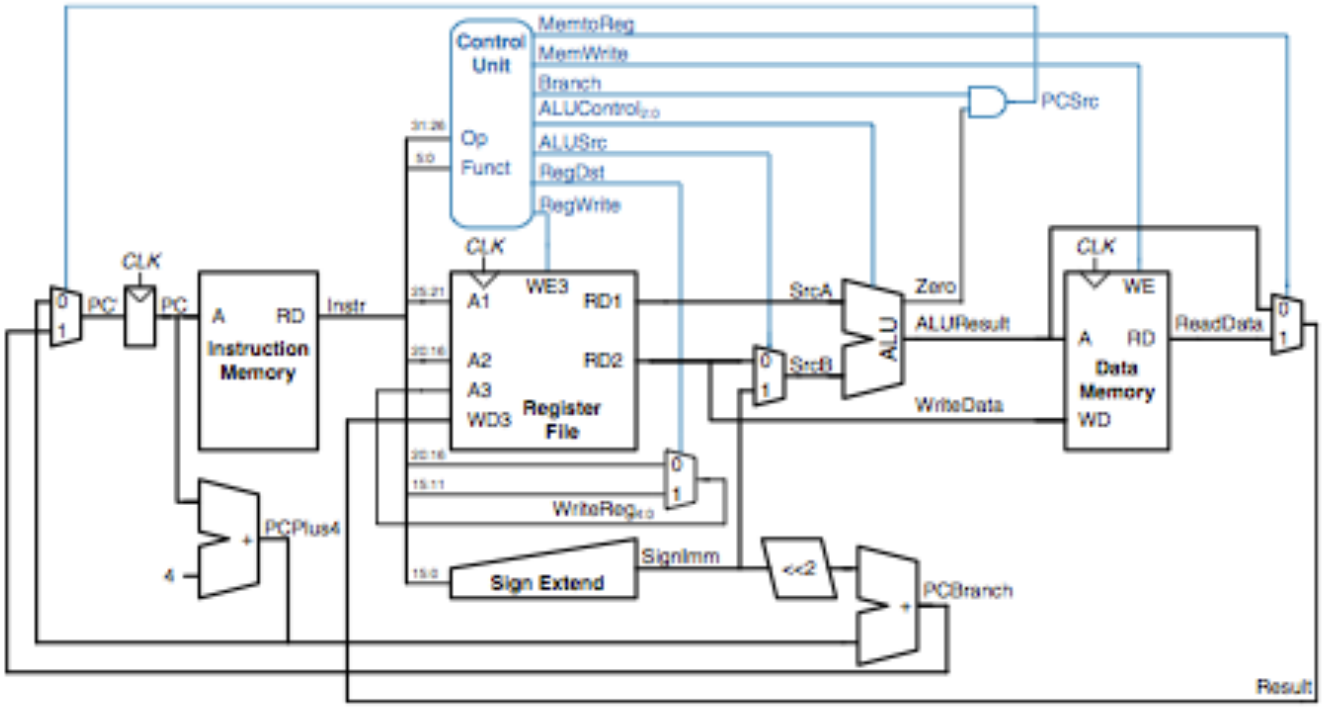


Figure 7.11 **Complete single-cycle MIPS processor**

Add support for the instruction "branch greater than zero" (bgtz, opcode 000111). List the control codes required and identify where additional hardware is required.

Key Questions

- Do I understand the datapath of the ALU in figure 5.15?
- What is the difference between data and control?
- How do I compute the performance of a processor?
- Do I understand the development of the single-cycle MIPS datapath (Figure 7.11)?

Extension: Improving Performance

$$Execution\ Time = \left(\#\ instructions\right)\left(\frac{cycles}{instruction}\right)\left(\frac{seconds}{cycle}\right) \quad (7.1)$$

- The equation above, introduced in section 7.2, describes performance on a processor.
- With a group around you, brainstorm ideas for improving performance on a processor.
- Link each idea to a particular term in the equation above.