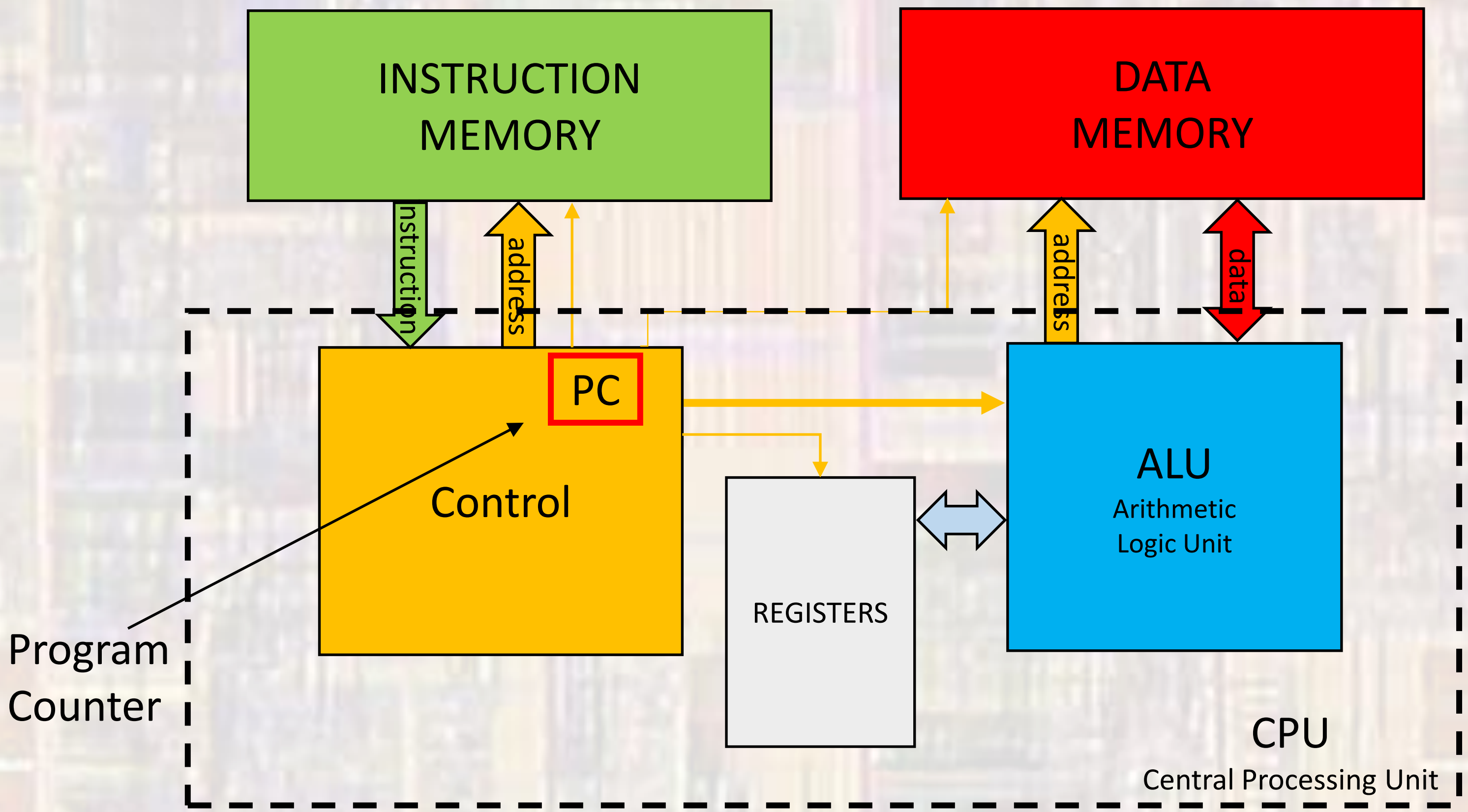
Linear Program Execution

Last updated 7/31/23

These slides describe linear program flow

Linear Program Execution

- Processor Architecture
  - Harvard – separate Instruction and Data memory paths



Linear Program Execution

- Instruction Sequencing
  - Program Counter (PC)
    - Register that holds the NEXT instruction memory location to be fetched
    - Provides the address for the instruction memory read
    - In linear program execution
      - The PC register is incremented each clock cycle
      - Incremented by the size of an instruction
      - e.g. for a 16 bit instruction word the PC would be incremented by 2
        - 0x1234 to 0x1236 since each instruction uses up 2 bytes

Linear Program Execution

- RISC Instruction set

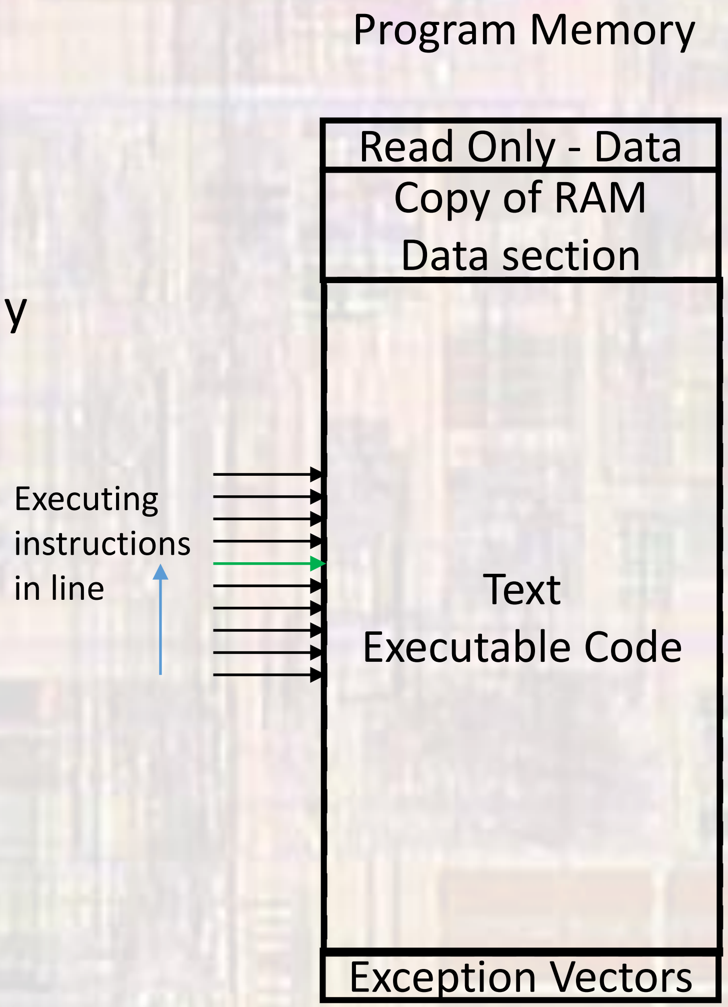
  - 2 basic types of instructions
    - Register based instructions
    - Memory instructions

  - Register Instructions
    - Only require access to the internal registers
      - Arithmetic
      - Logical
      - Control

  - Memory Operations
    - Read or write to memory/registers

Linear Program Execution

- Instruction Sequencing
  - Program control
    - Linear flow – increment PC normally



Linear Program Execution

- Instruction Execution
  - 6 possible steps for each instruction
    - 2 required, 4 optional

	Fetch		get next instruction from instruction memory
	Decode		determine what the instruction is
depend on instruction	Execute		if necessary – do what the instruction requires
	Load	Mem	if necessary – get value from data memory
	Store		if necessary – place value in data memory
	Write Back		if necessary – store result in register

- After the Fetch – increment the PC to point to the next instruction

Linear Program Execution

- 1 line of code - complete

The compiler has assigned
b to memory location 4000
c to memory location 4004
a to memory location 4008

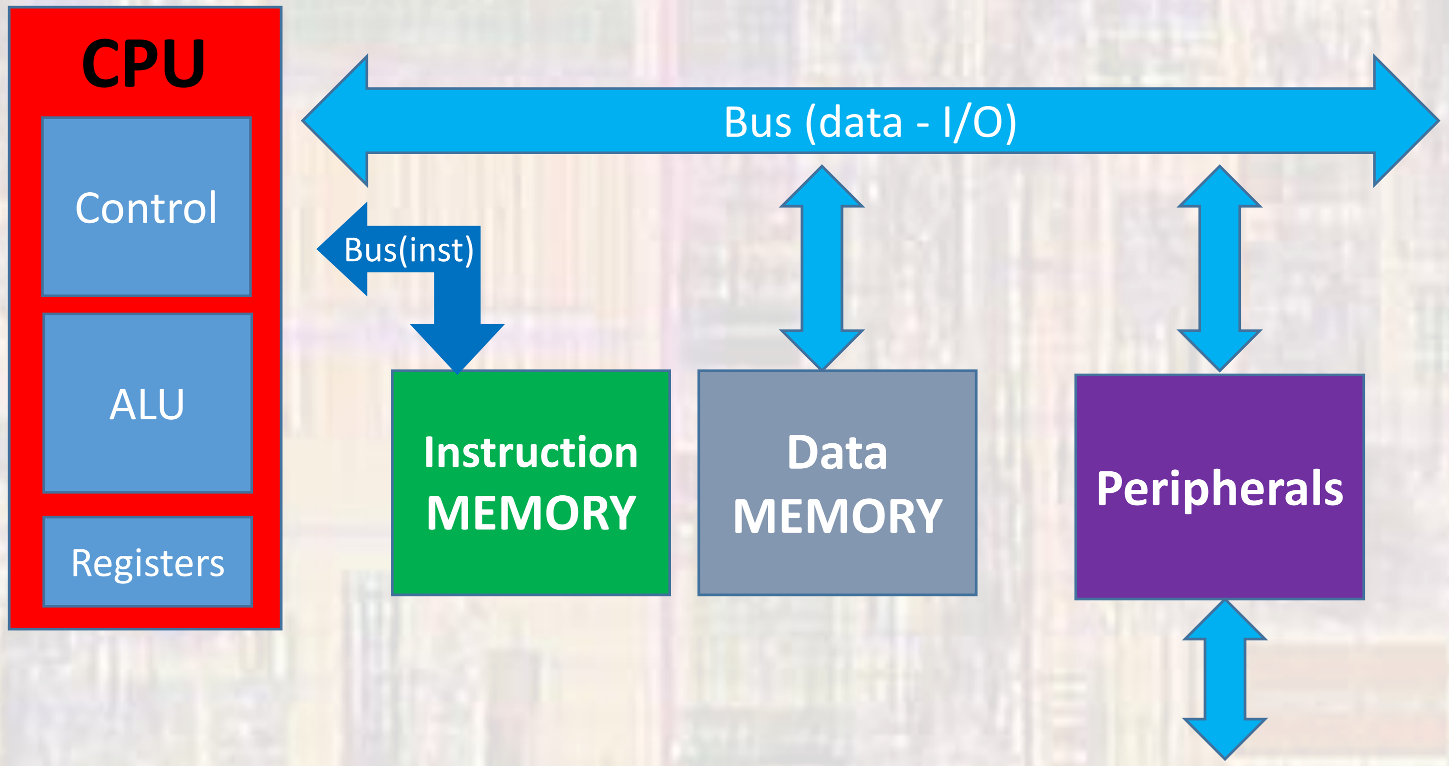
a = b + c;

The compiler turned the single line into 7 instructions

Mem loc	Instruction	Encoding	action
1000	ldi R1, 4000	1001 0000	Load loc for B into R1
1002	ld R2, R1	0001 0001	Put value at loc for B in R2 ld R2, mem(R1)
1004	ldi R1, 4004	1001 0010	Load loc for C into R1
1006	ld R3, R1	0001 0010	Put value at loc for C in R3 ld R3, mem(R1)
1008	add R2, R3, R4	0x27	R4 <- R2 + R3
100A	ldi R1, 4008	0x84	Load loc for A into R1
100C	st R1, R4	0x21	Put value of R4 into loc for A st mem(R1), R4

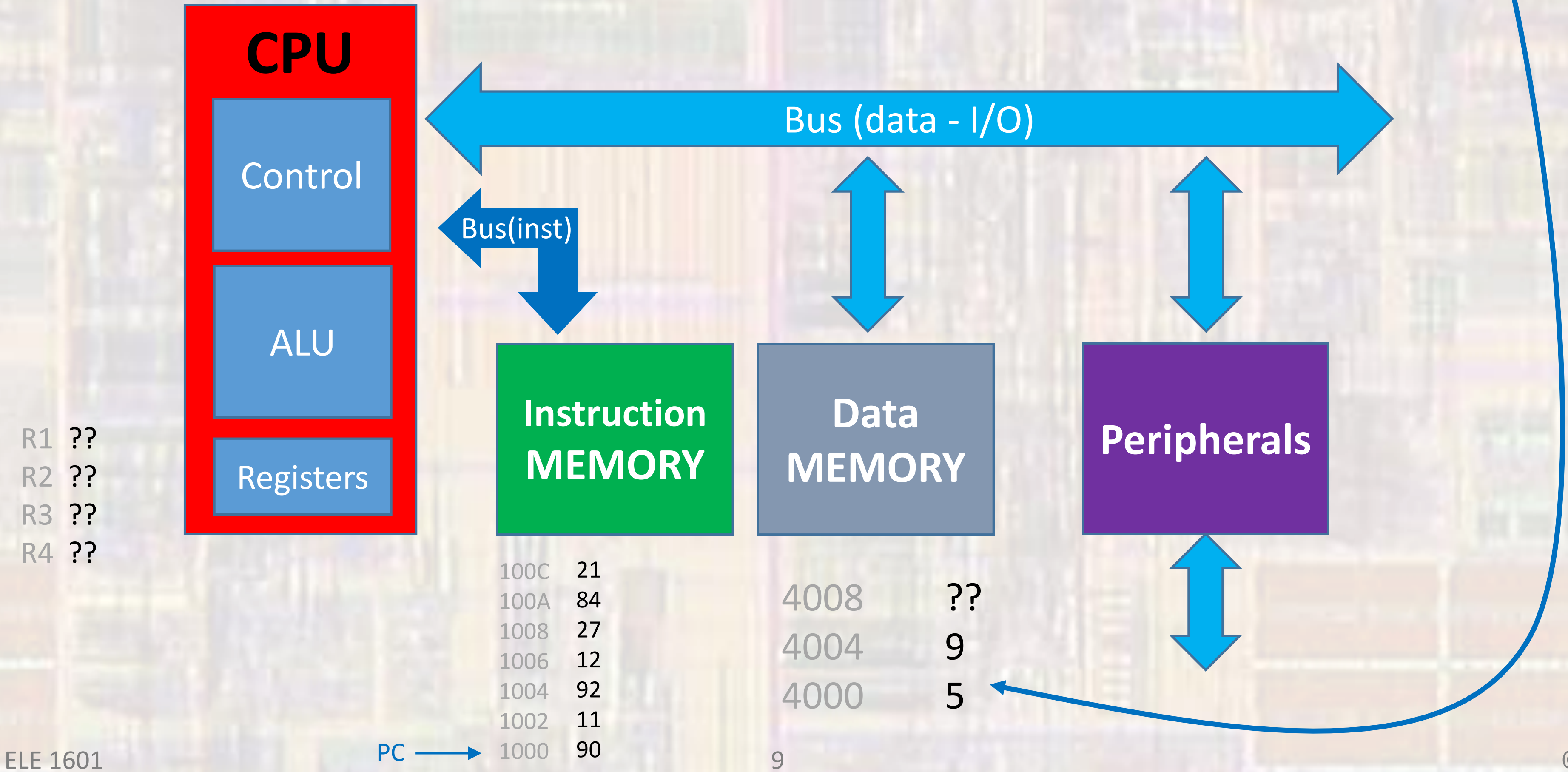
Linear Program Execution

- Simplified Block Diagram



Linear Program Execution

- Status
  - Data locations filled by previous commands
  - PC currently pointing to Instruction memory location 1000



R1	??
R2	??
R3	??
R4	??

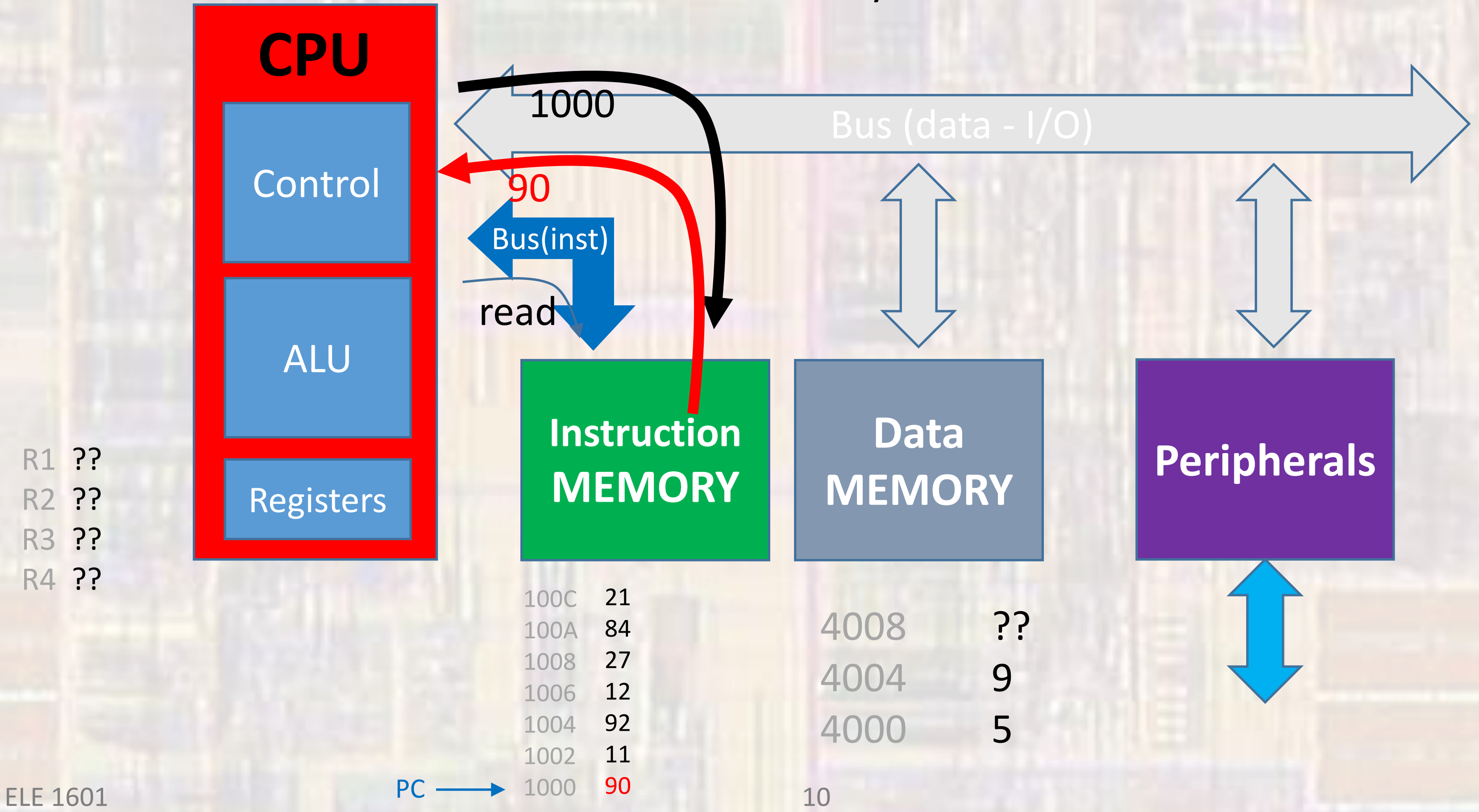
100C	21
100A	84
1008	27
1006	12
1004	92
1002	11
1000	90

4008	??
4004	9
4000	5

Linear Program Execution

- First Instruction (fetch)

Control puts a memory location from the PC (1000) on the address bus along with a read signal
Instruction memory returns the value at that location (90)



R1	??
R2	??
R3	??
R4	??

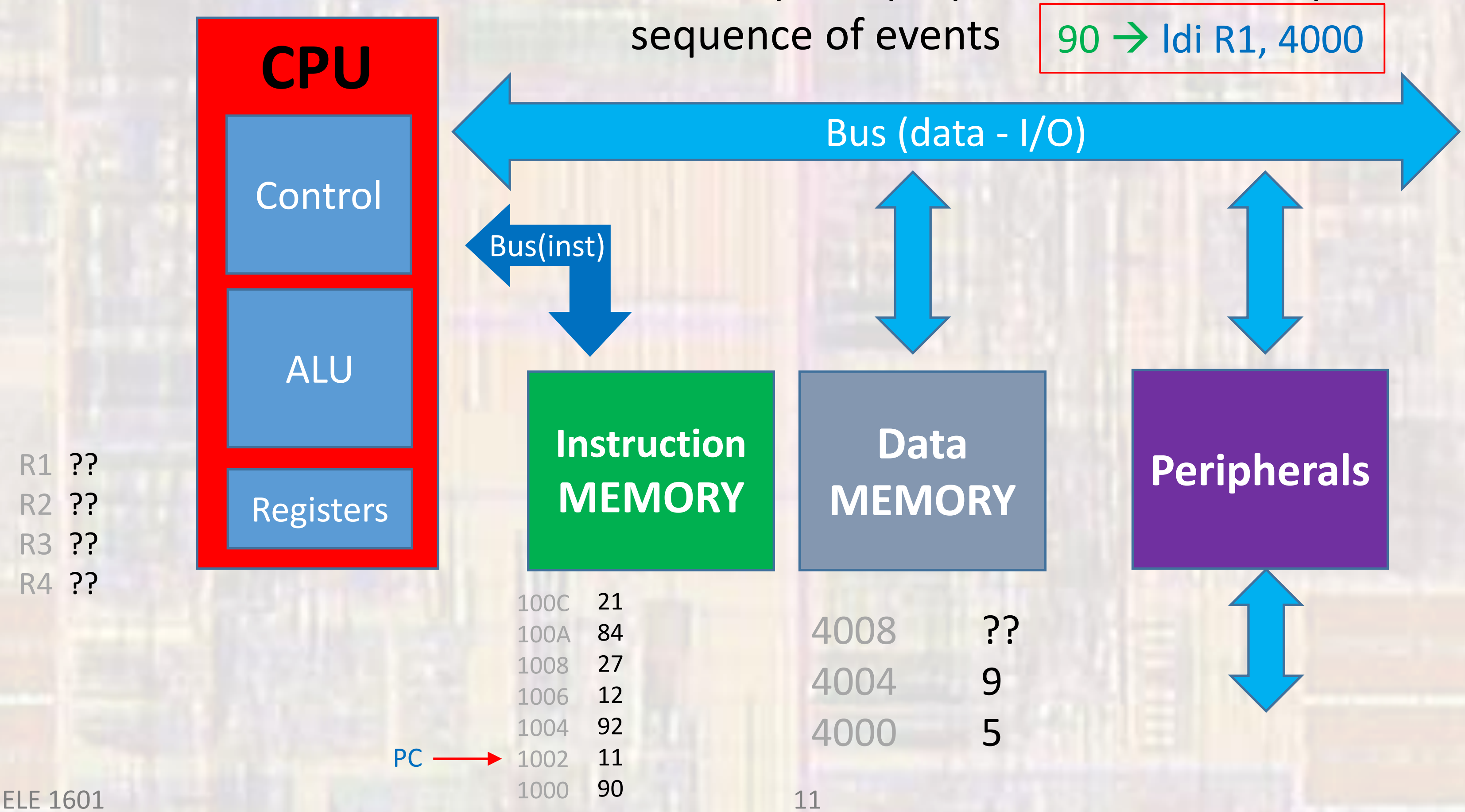
100C	21
100A	84
1008	27
1006	12
1004	92
1002	11
1000	90

4008	??
4004	9
4000	5

Linear Program Execution

- First Instruction (decode)

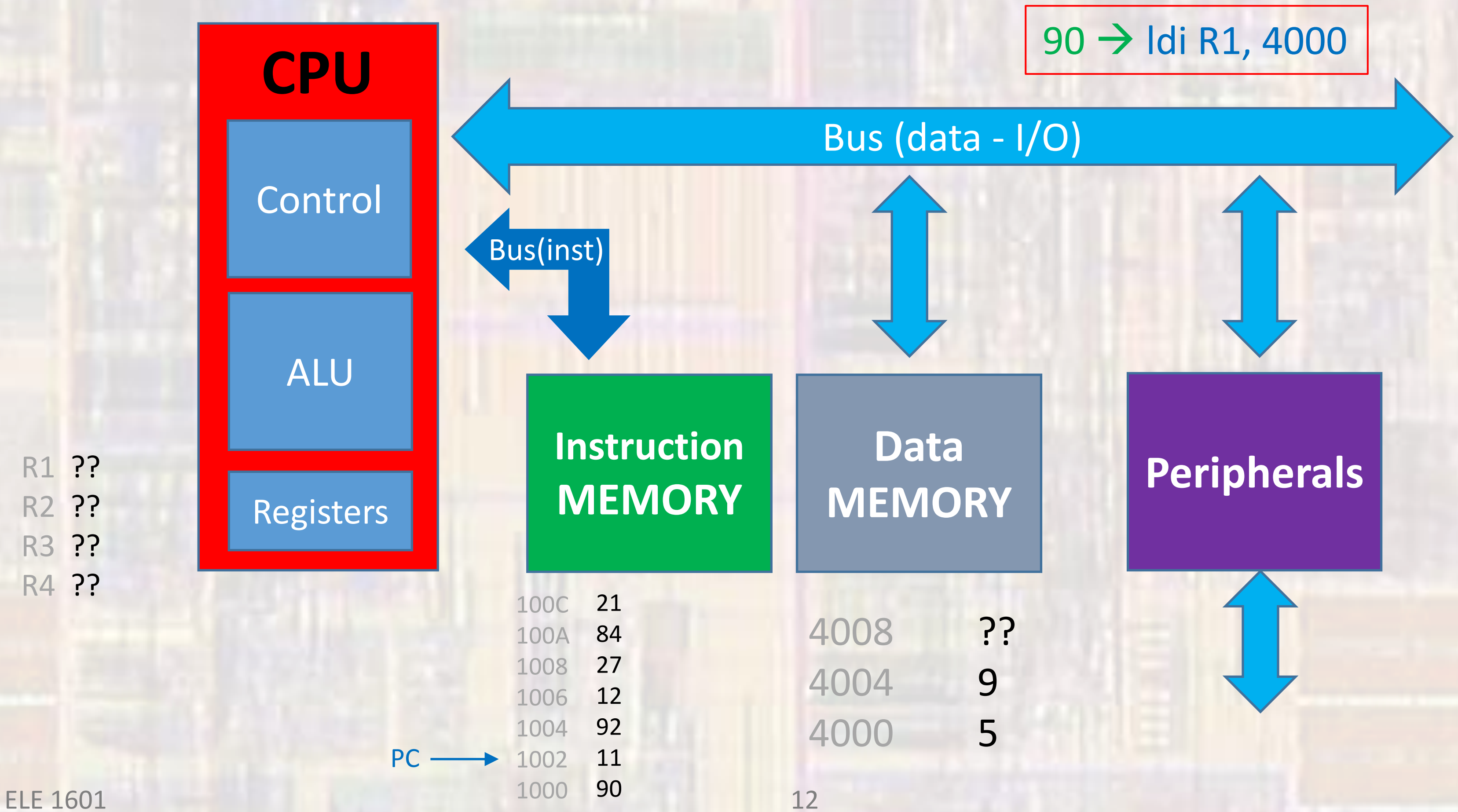
Control decodes the word returned by the memory and prepares to execute a pre-defined sequence of events



Linear Program Execution

- First Instruction (execute)

Does nothing for this instruction



R1	??
R2	??
R3	??
R4	??

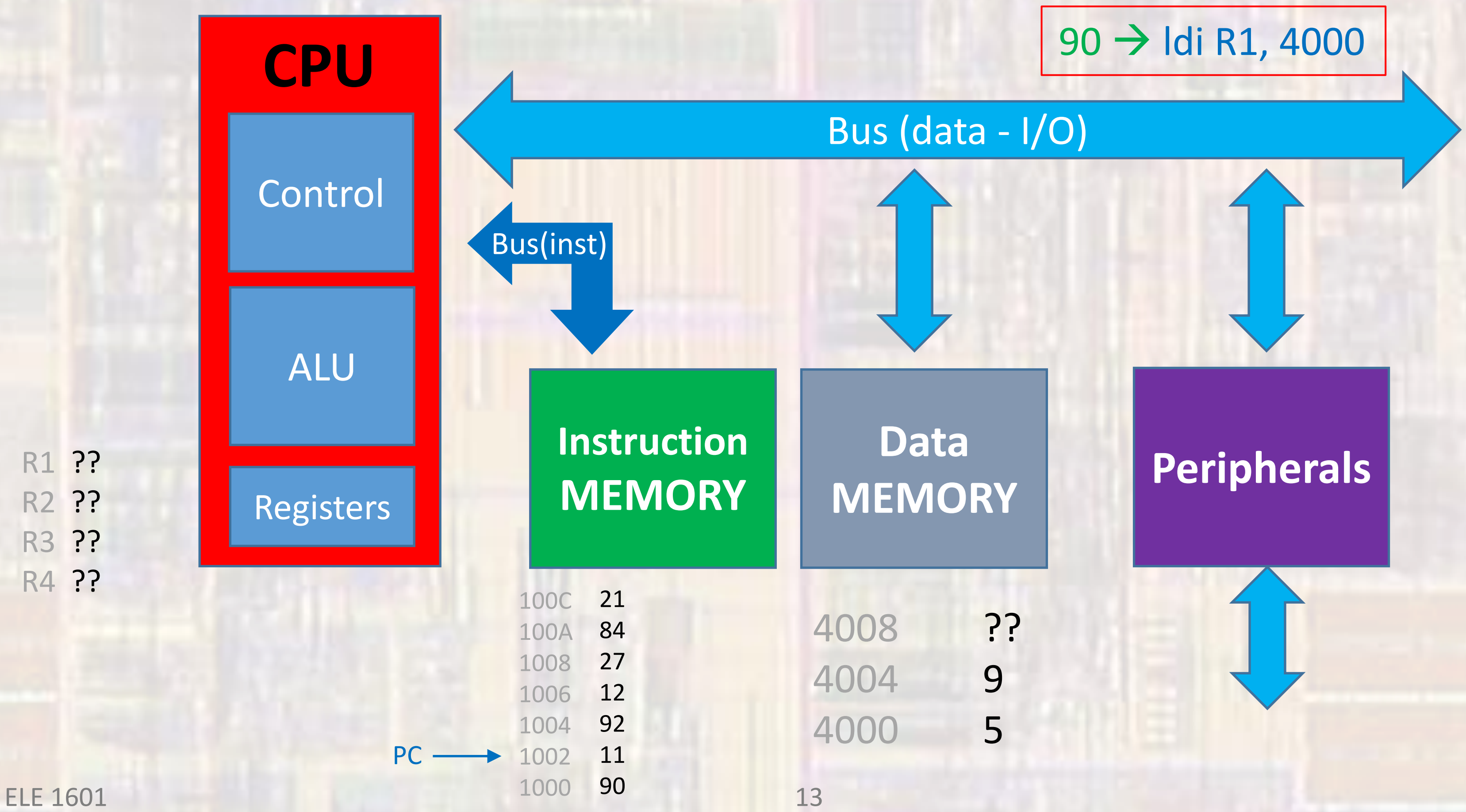
100C	21
100A	84
1008	27
1006	12
1004	92
1002	11
1000	90

4008	??
4004	9
4000	5

Linear Program Execution

- First Instruction (mem)

Does nothing for this instruction



R1	??
R2	??
R3	??
R4	??

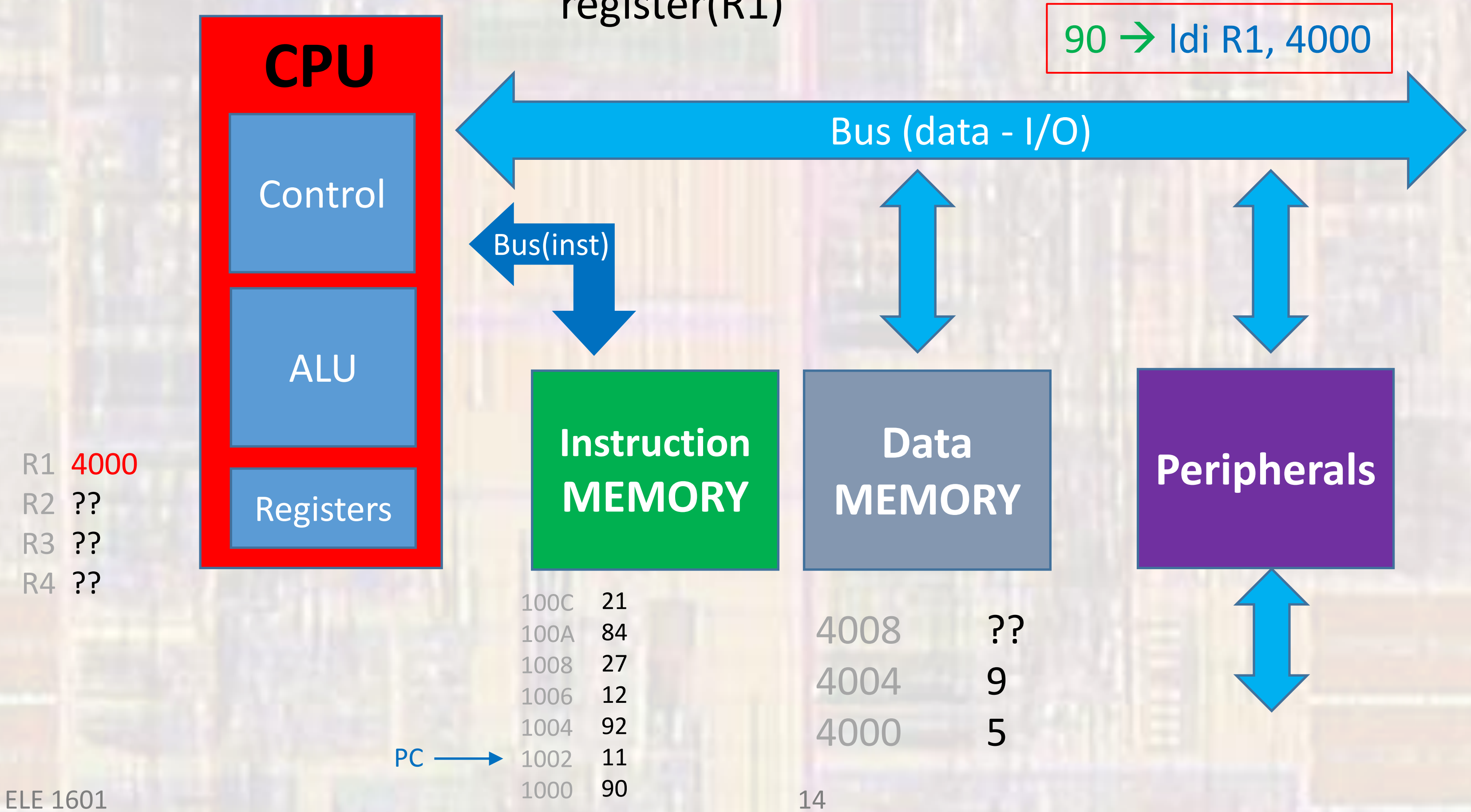
100C	21
100A	84
1008	27
1006	12
1004	92
1002	11
1000	90

4008	??
4004	9
4000	5

Linear Program Execution

- First Instruction (write back)

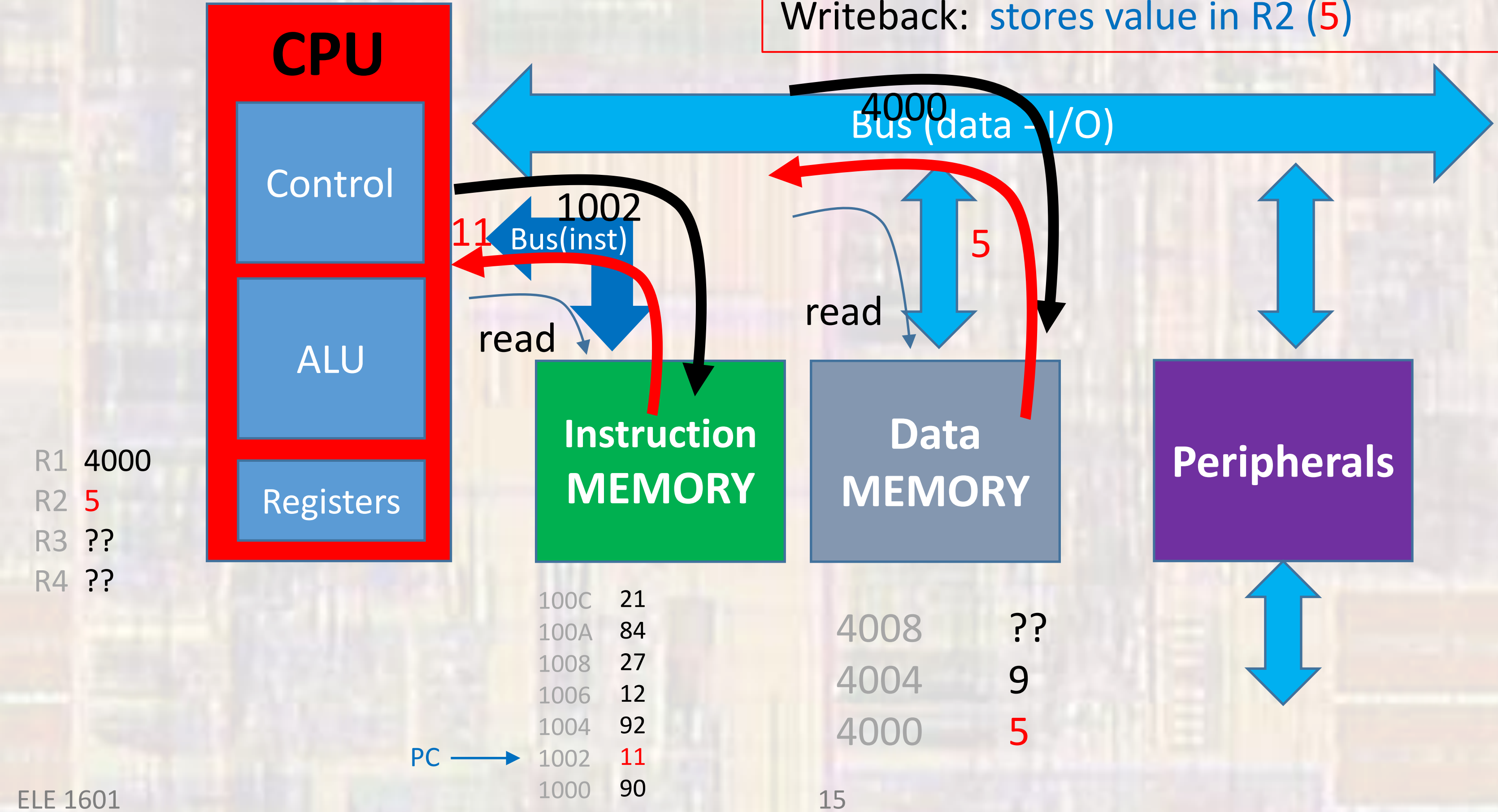
CPU writes the value(4000) back into a register(R1)



Linear Program Execution

- New Fetch

Fetch:	→ 1002 ← 11
Decode:	11 → ld R2, mem(R1) ld R2, mem(R1)
Execute:	idle
MEM:	value at location in R1(4000) ← (5)
Writeback:	stores value in R2 (5)



R1	4000
R2	5
R3	??
R4	??

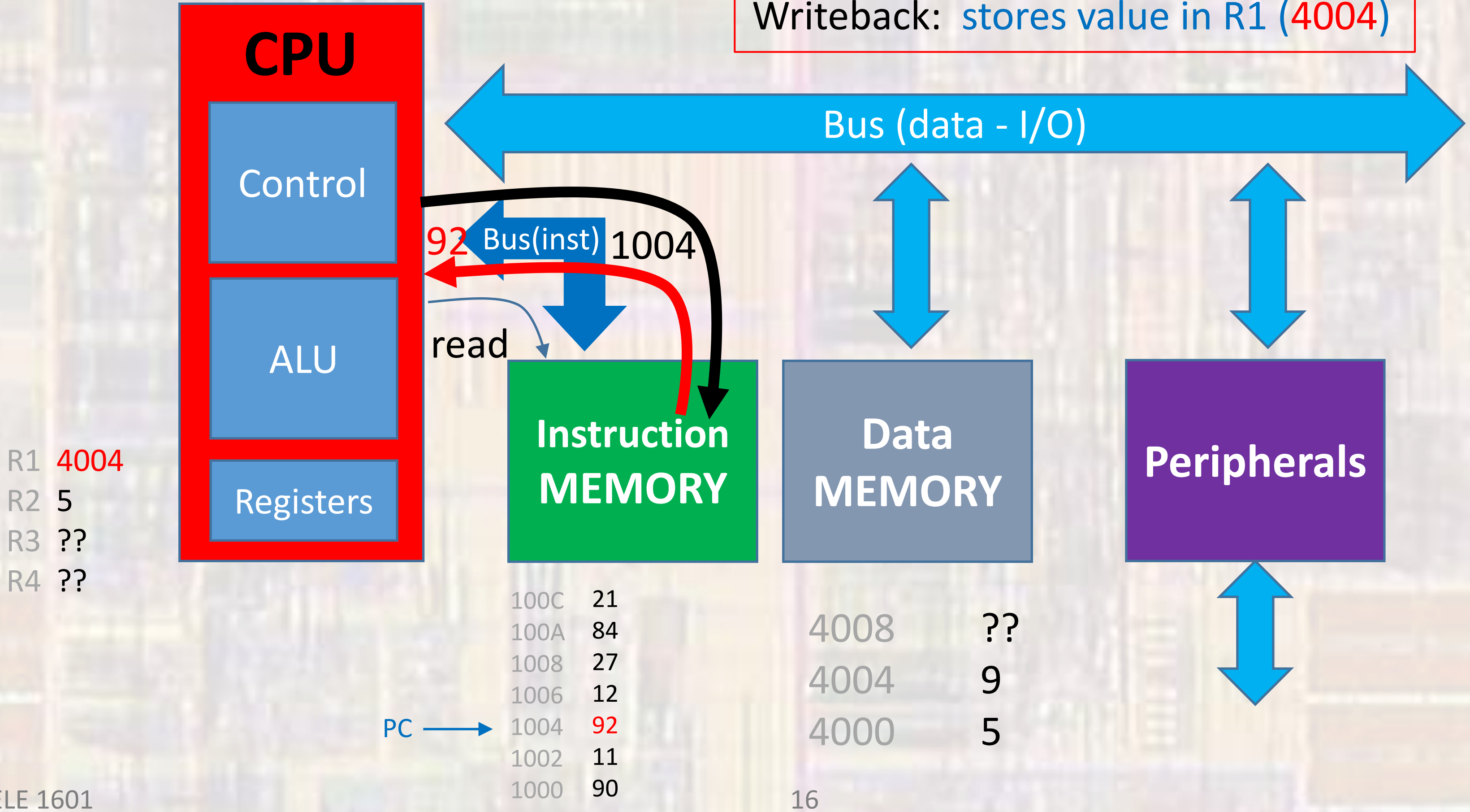
100C	21
100A	84
1008	27
1006	12
1004	92
1002	11
1000	90

4008	??
4004	9
4000	5

Linear Program Execution

- New Fetch

Fetch:	→ 1004 ← 92
Decode:	92 → ldi R1, 4004
Execute:	idle
MEM:	idle
Writeback:	stores value in R1 (4004)



R1	4004
R2	5
R3	??
R4	??

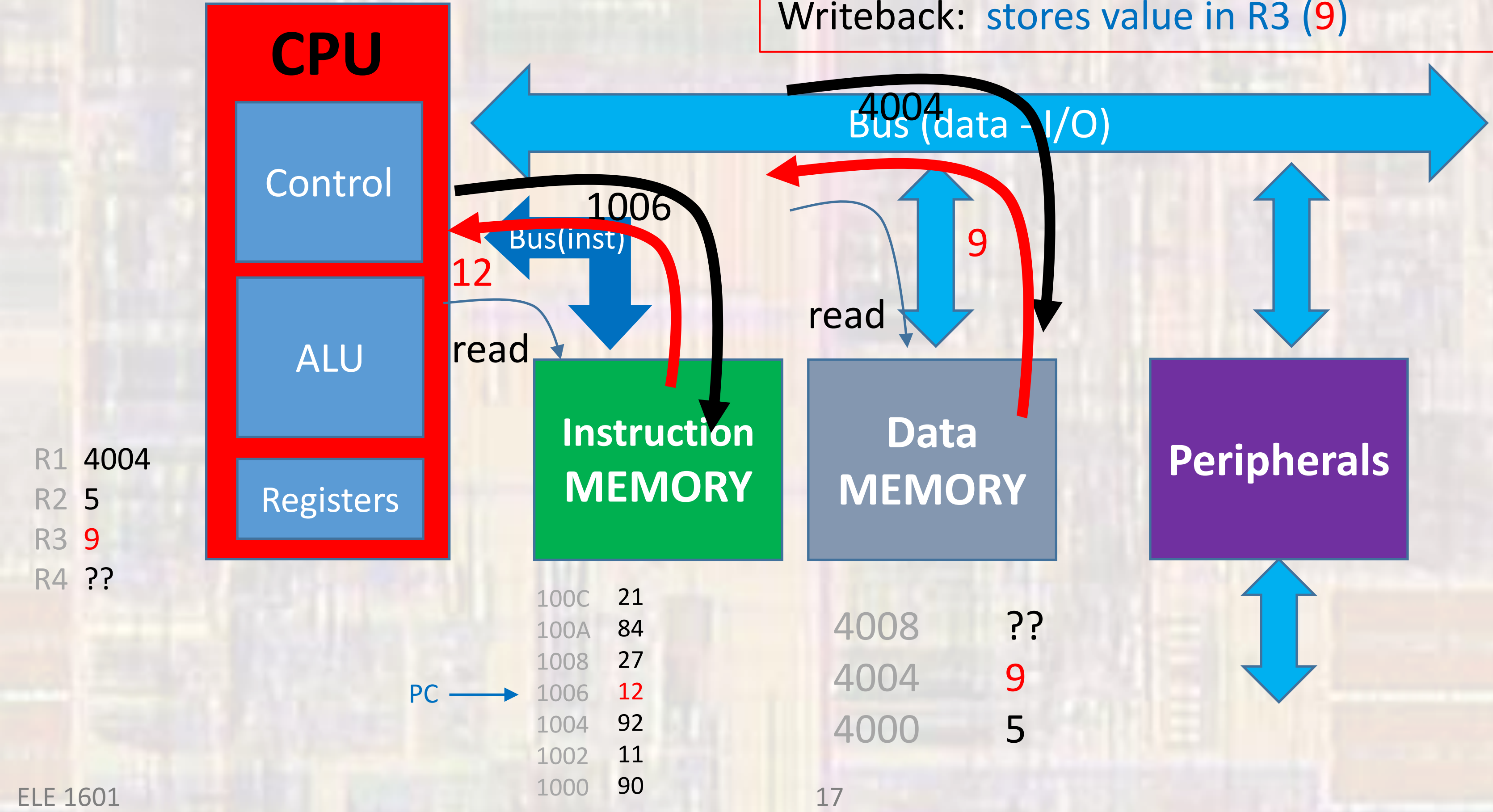
100C	21
100A	84
1008	27
1006	12
1004	92
1002	11
1000	90

4008	??
4004	9
4000	5

Linear Program Execution

- New Fetch

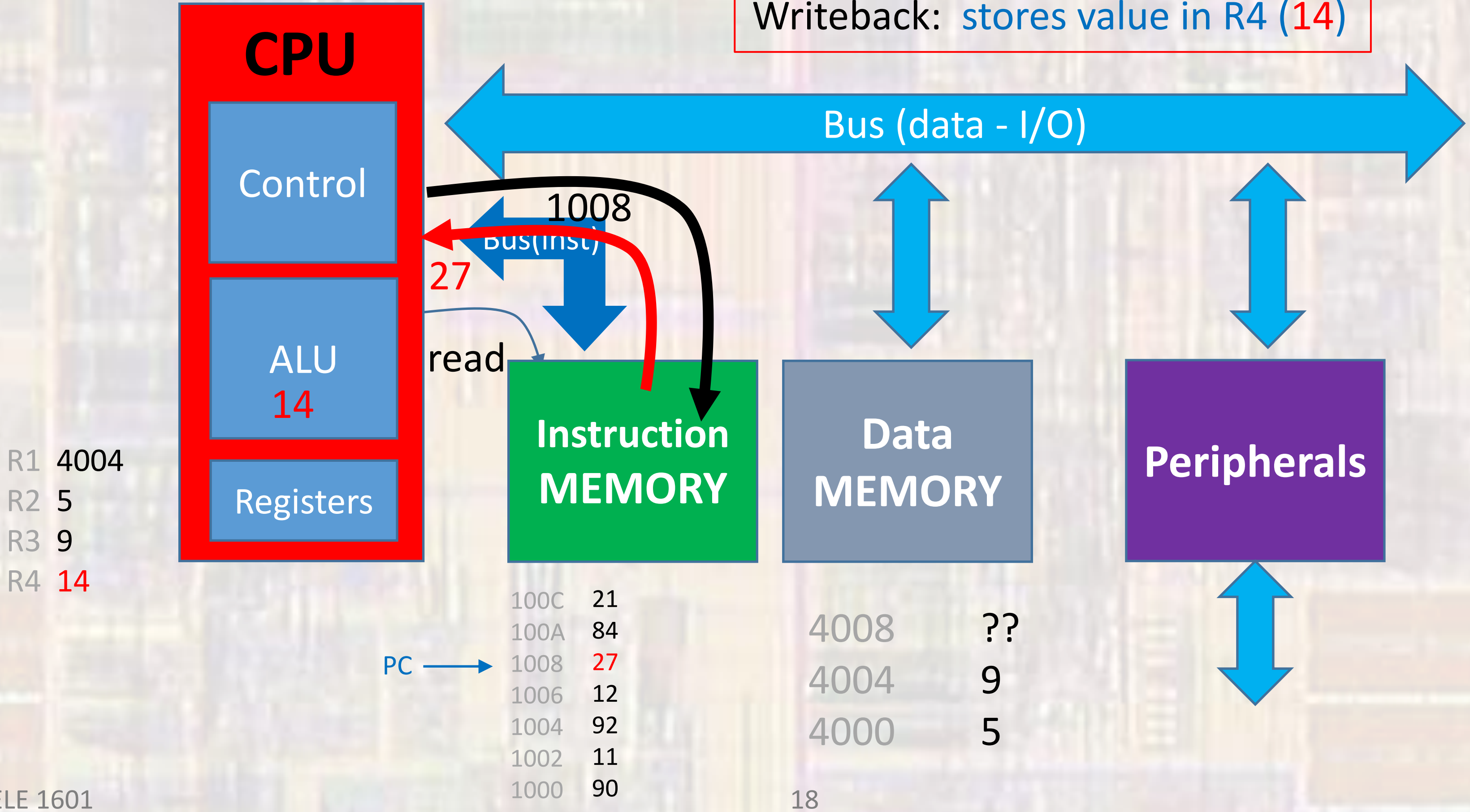
Fetch: → 1006 ← 12
Decode: 12 → ld R3, R1 ld R3, mem(R1)
Execute: idle
MEM: value at location in R1(4004) ← (9)
Writeback: stores value in R3 (9)



Linear Program Execution

- New Fetch

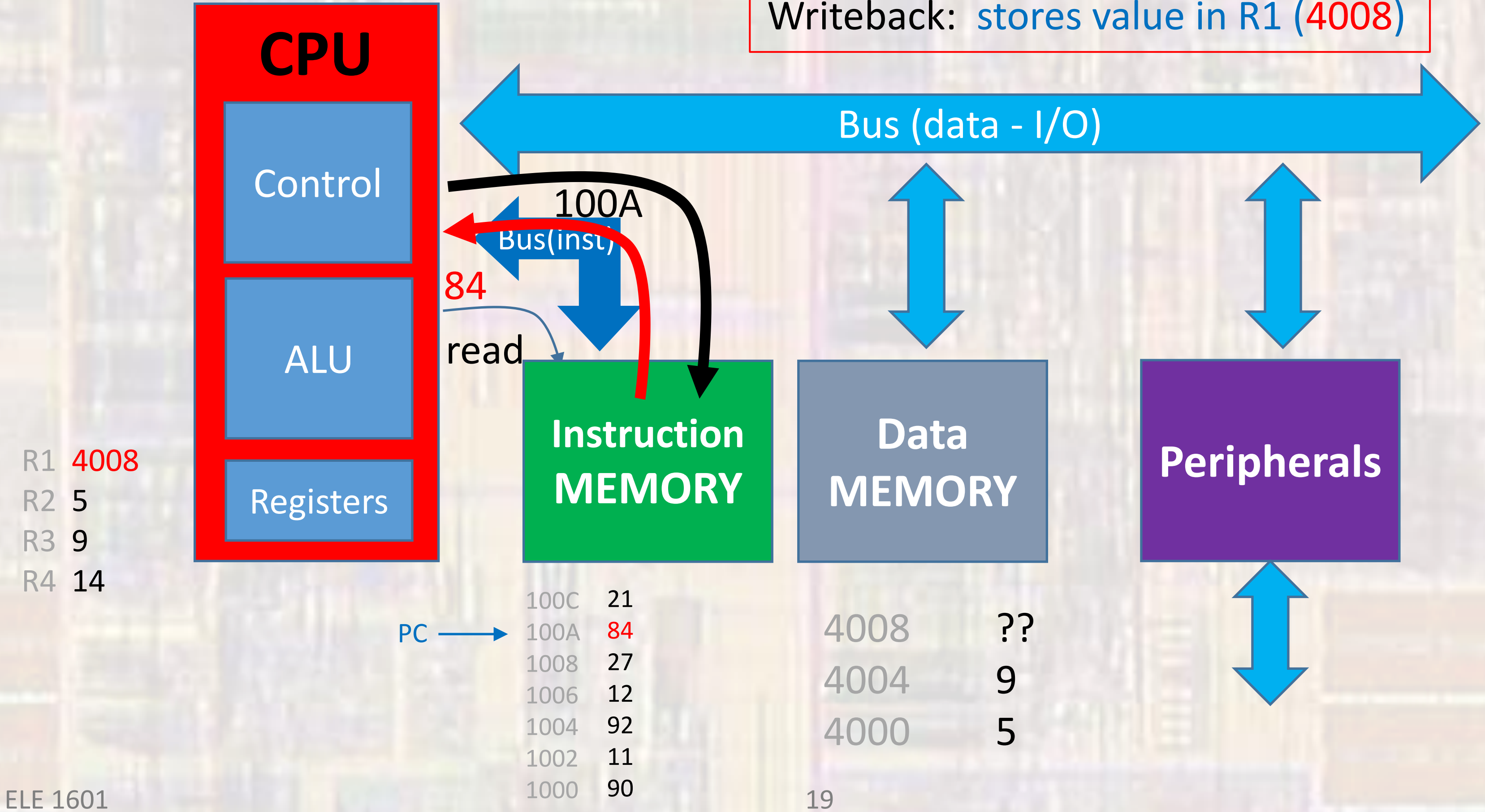
Fetch: → 1008 ← 27
Decode: 27 → add R2, R3, R4
Execute: adds R2 + R3 → 14
MEM: idle
Writeback: stores value in R4 (14)



Linear Program Execution

- New Fetch

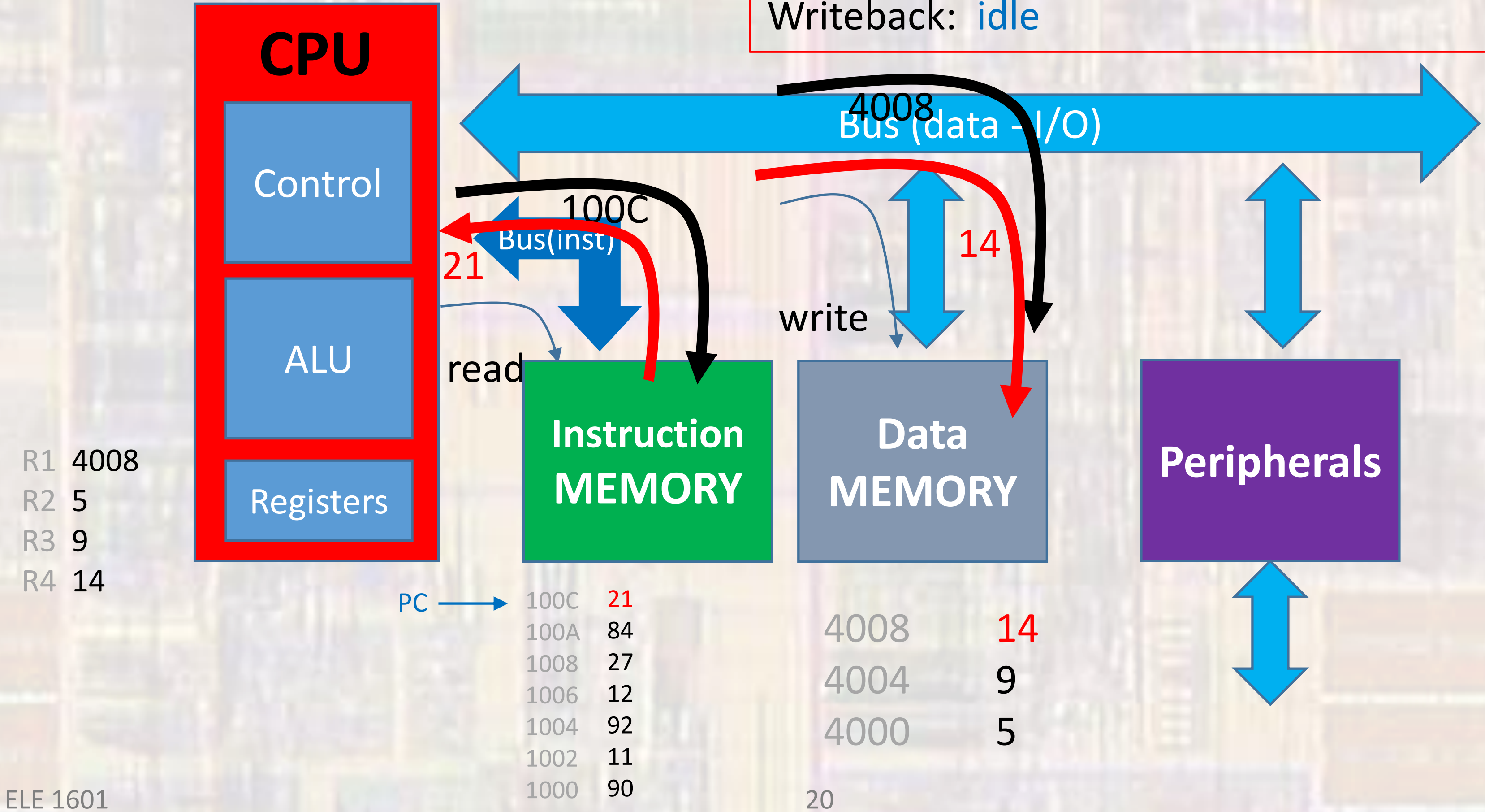
Fetch: → 100A ← 84
Decode: 84 → ldi R1, 4008
Execute: idle
MEM: idle
Writeback: stores value in R1 (4008)



Linear Program Execution

- New Fetch

Fetch:	→ 100C ← 21
Decode:	21 → st R1, R4 st mem(R1), R4
Execute:	idle
MEM:	R4(14) → location in R1(4008)
Writeback:	idle



R1	4008
R2	5
R3	9
R4	14

100C	21
100A	84
1008	27
1006	12
1004	92
1002	11
1000	90

4008	14
4004	9
4000	5