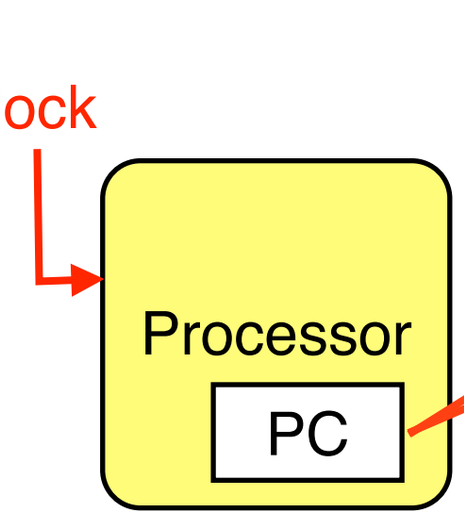
Processor Design – Single Cycle Processor

Hung-Wei Tseng

Recap: the stored-program computer

- Store instructions in memory
- The program counter (PC) controls the execution



instruction memory

```
120007a30:   0f00bb27   ldah   gp,15(t12)
120007a34:   509cbd23   lda    gp,-25520(gp)
120007a38:   00005d24   ldah   t1,0(gp)
120007a3c:   0000bd24   ldah   t4,0(gp)
120007a40:   2ca422a0   ldl    t0,-23508(t1)
120007a44:   130020e4   beq    t0,120007a94
120007a48:   00003d24   ldah   t0,0(gp)
120007a4c:   2ca4e2b3   stl    zero,-23508(t1)
120007a50:   0004ff47   clr    v0
120007a54:   28a4e5b3   stl    zero,-23512(t4)
120007a58:   20a421a4   ldq    t0,-23520(t0)
120007a5c:   0e0020e4   beq    t0,120007a98
120007a60:   0204e147   mov    t0,t1
120007a64:   0304ff47   clr    t2
120007a68:   0500e0c3   br     120007a80
```

How many of these?

Instruction Count!

How long is it take to execution each of these?

Cycles per instruction * cycle time

Recap: Performance Equation

$$\text{Execution Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

- ET = IC * CPI * Cycle Time
- IC (Instruction Count)
  - ISA, compiler, algorithm, programming language, **programmer**
- CPI (Cycles Per Instruction)
  - Machine implementation, microarchitecture, compiler, application, algorithm, programming language, **programmer**
- Cycle Time (Seconds Per Cycle)
  - Process technology, microarchitecture, ISA, **programmer**

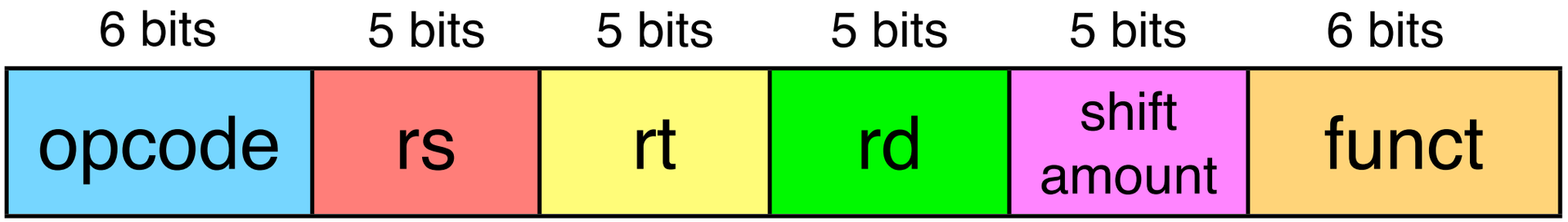
We demonstrated these

We learned about this

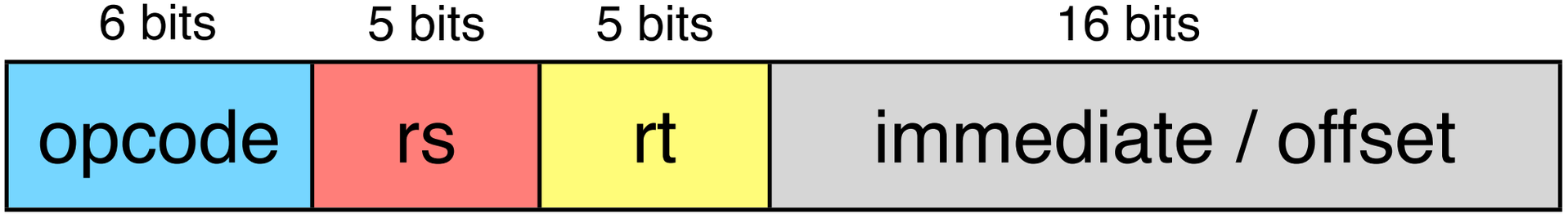
We will be talking about this!

Recap: MIPS ISA

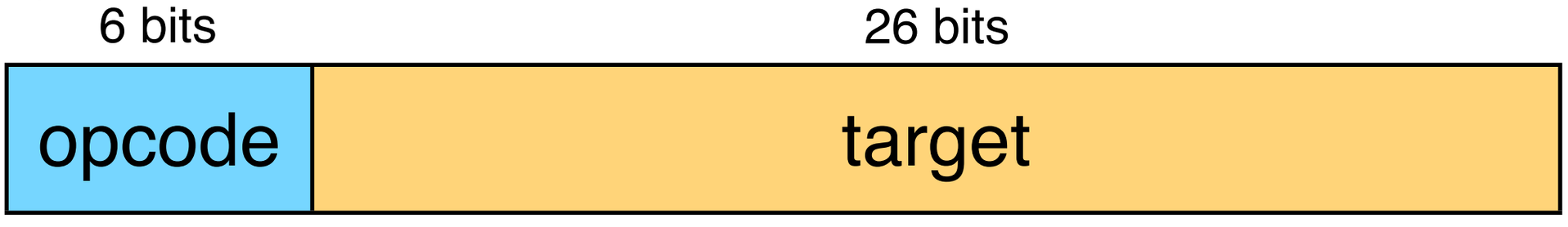
- R-type: add, sub, and etc...



- I-type: addi, lw, sw, beq, and etc...

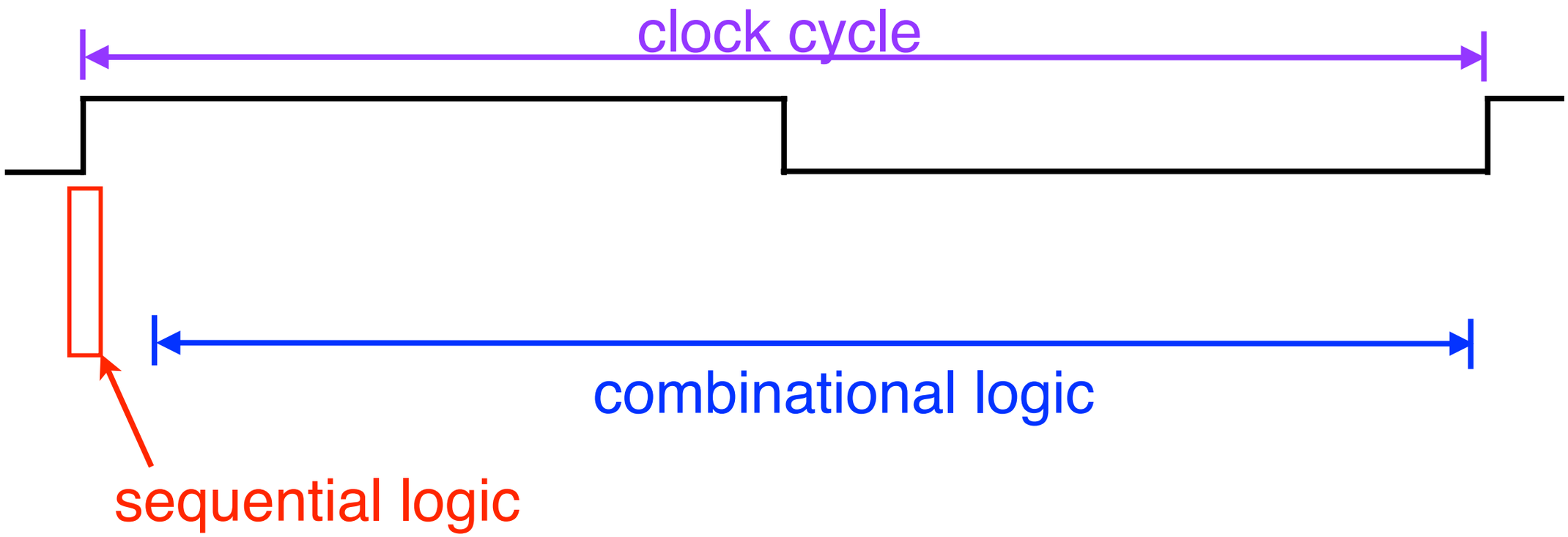


- J-type: j, jal, and etc...



Recap from CSE140: Clock

- A hardware signal defines when data is valid and stable
  - Think about the clock in real life!
- We use edge-triggered clocking
  - Values stored in the sequential logic is updated only on a clock edge



Announcement

- Reading quizzes for 4.5-4.9 due tomorrow
- No new reading quizzes until midterm (oh! yeah~~~)
- Homework 2 due next Monday

Outline

- Implementing a Single-cycle MIPS processor

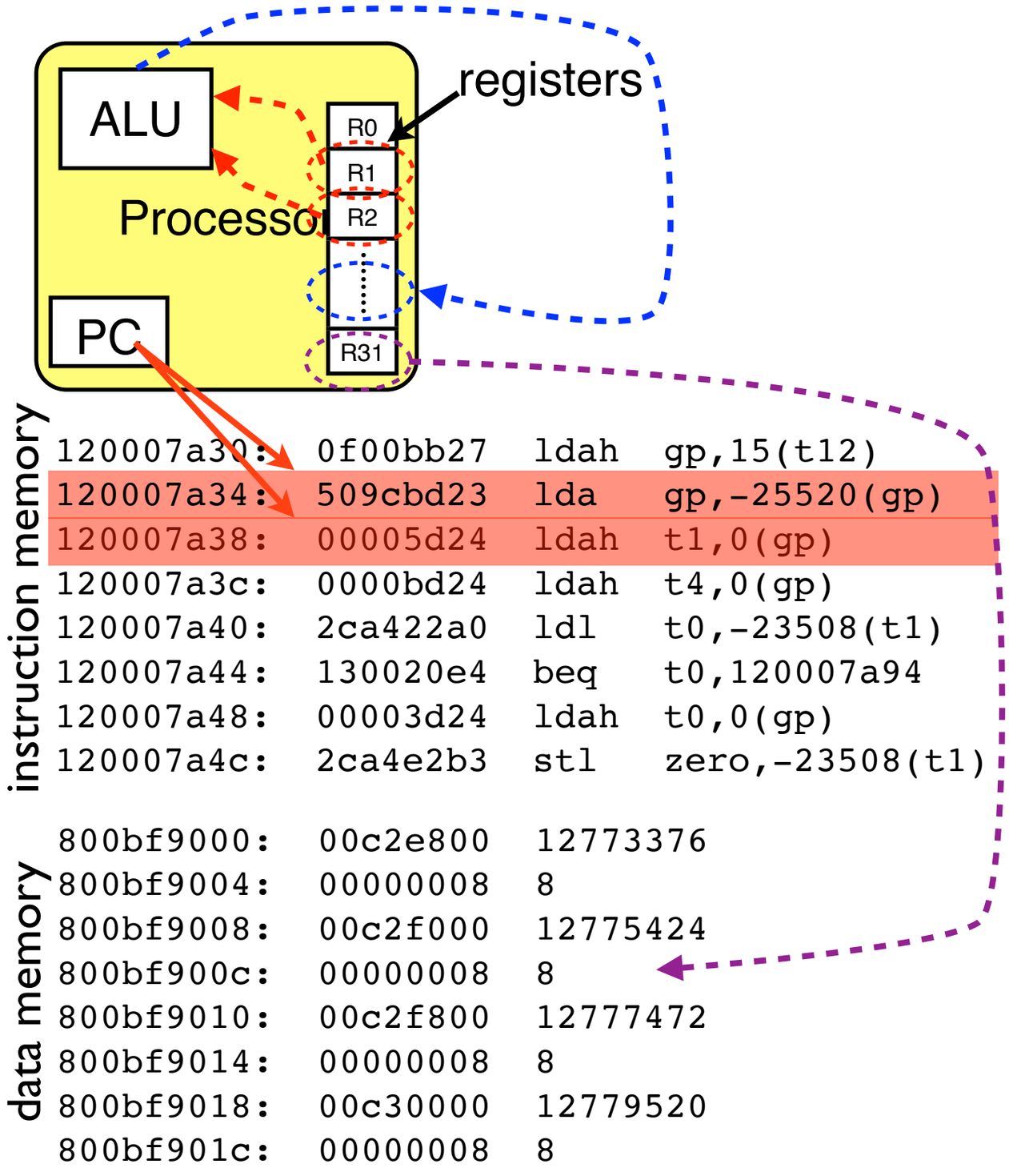
Designing a simple MIPS processor

- Support MIPS ISA in hardware
  - Design the datapath: add and connect all the required elements in the right order
  - Design the control path: control each datapath element to function correctly.
- Starts from designing a single cycle processor
  - Each instruction takes exactly one cycle to execute

Basic steps of execution

- Instruction fetch: where?
  instruction memory
- Decode:
  - What's the instruction?
  - Where are the operands?
    registers
- Execute **ALUs**
- Memory access
  - Where is my data?
    data memory
- Write back
  - Where to put the result

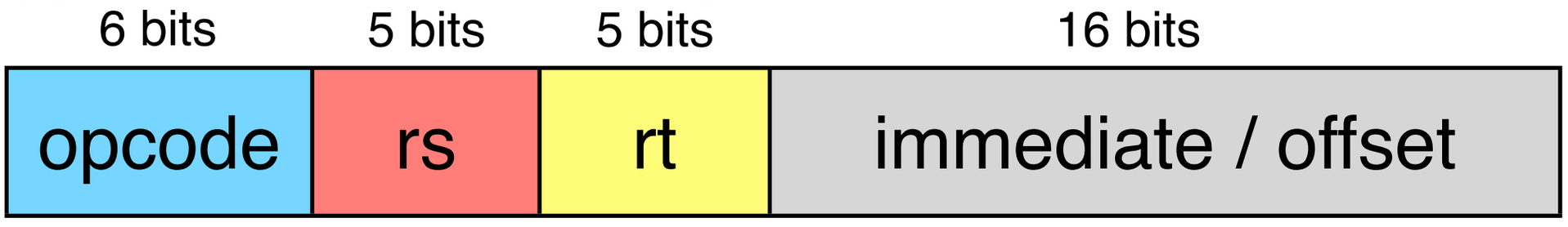
  registers
- Determine the next PC



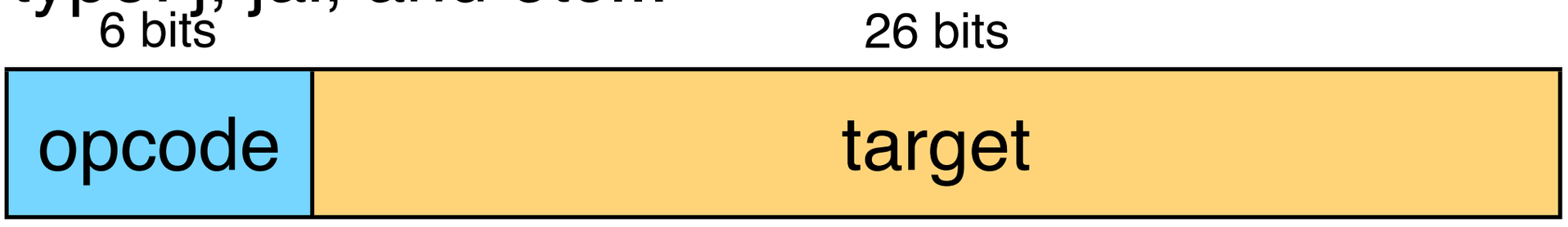
Recap: MIPS ISA

- R-type: add, sub, and etc...

- I-type: addi, lw, sw, beq, and etc...



- J-type: j, jal, and etc...

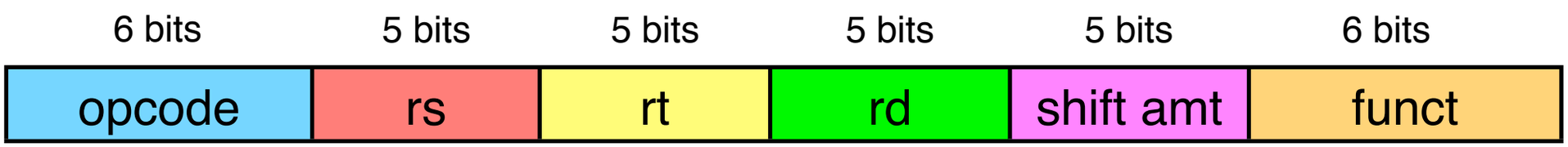


Implementing an R-type instruction

- How many of the following datapath elements is necessary for an R-type instruction?
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. 1
B. 2
C. 3
D. 4
E. 5

instruction = MEM[PC]
REG[rd] = REG[rs] op REG[rt]
PC = PC + 4

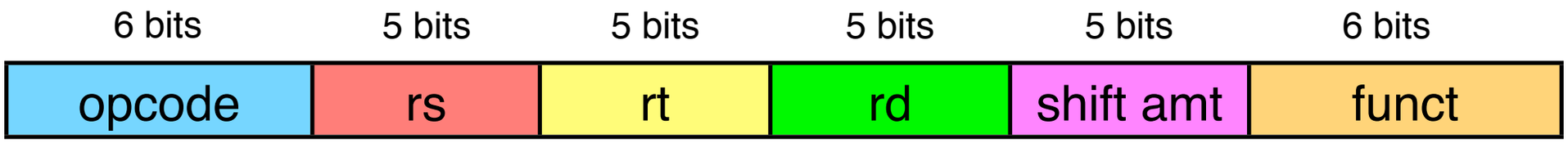


Implementing an R-type instruction

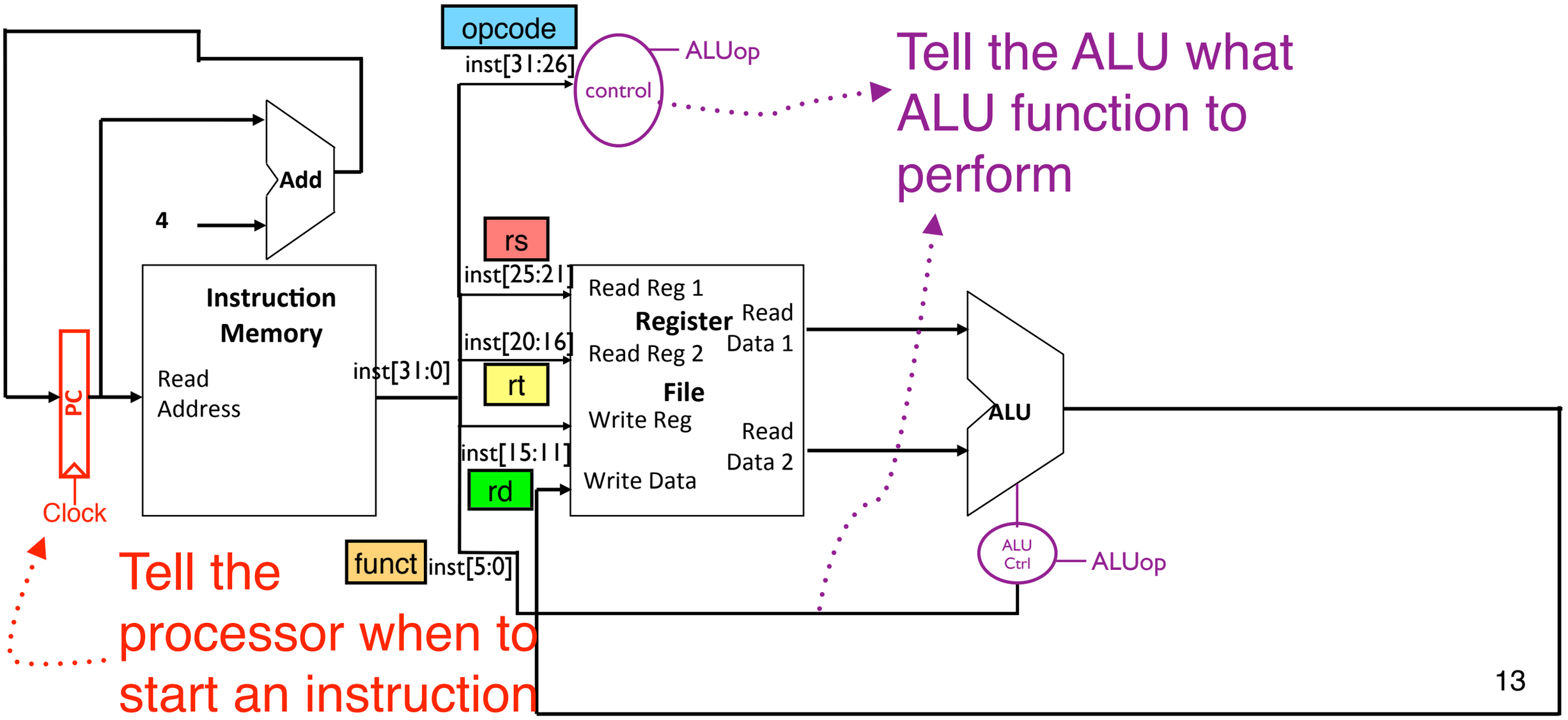
- What's right order of accessing the datapath elements for an R-type instruction? (only count the first time touching it)
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. I, III, V, IV
B. IV, I, III, V
C. I, V, III, IV
D. IV, V, I, III
E. none of the above

Implementing an R-type instruction



instruction = MEM[PC]
REG[rd] = REG[rs] op REG[rt]
PC = PC + 4

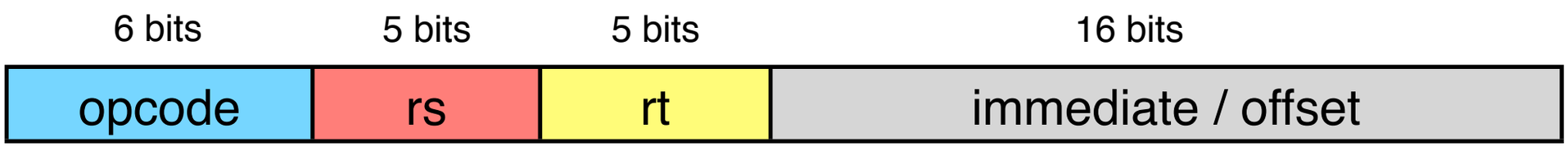


Implementing a load instruction

- How many of the following datapath elements is necessary for a load instruction?
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. 1
B. 2
C. 3
D. 4
E. 5

instruction = MEM[PC]
REG[rt] = MEM[signext(immediate) + REG[rs]]
PC = PC + 4

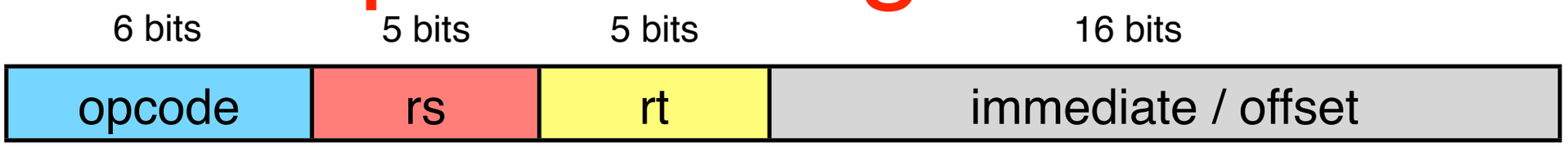


Implementing a load instruction

- What's right order of accessing the datapath elements for a load instruction? (only count the first time touching it)
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. IV, I, III, V, II
B. IV, I, III, II, V
C. IV, I, V, II, III
D. IV, I, II, V, III
E. none of the above

Implementing a load instruction



instruction = MEM[PC]
REG[rt] = MEM[signext(immediate) + REG[rs]]
PC = PC + 4

Set different control signals for different types of instructions

Set to 1 if it's a load

Set to 1 if it's a load

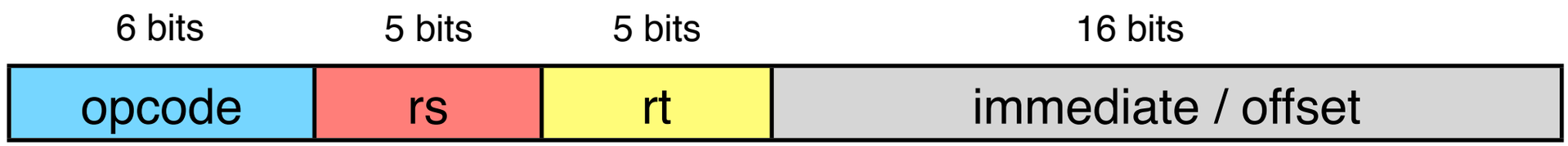
Set to 0 if it's a load

Implementing a store instruction

- How many of the following datapath elements is necessary for a store instruction?
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. 1
B. 2
C. 3
D. 4
E. 5

instruction = MEM[PC]
MEM[signext(immediate) + REG[rs]] = REG[rt]
PC = PC + 4

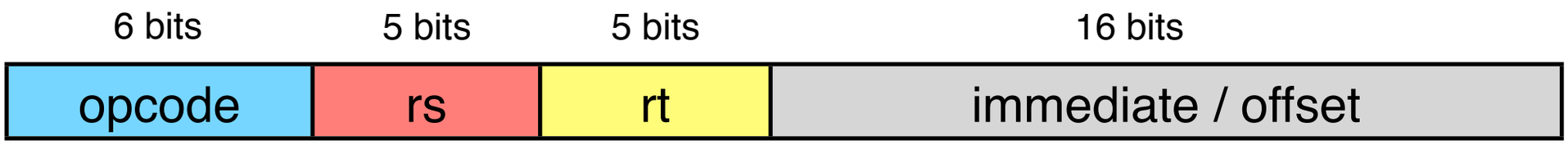


Implementing a store instruction

- What's right order of accessing the datapath elements for a store instruction? (only count the first time touching it)
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. IV, I, III, V, II
B. IV, I, III, II, V
C. IV, I, V, II, III
D. IV, I, II, V, III
E. none of the above

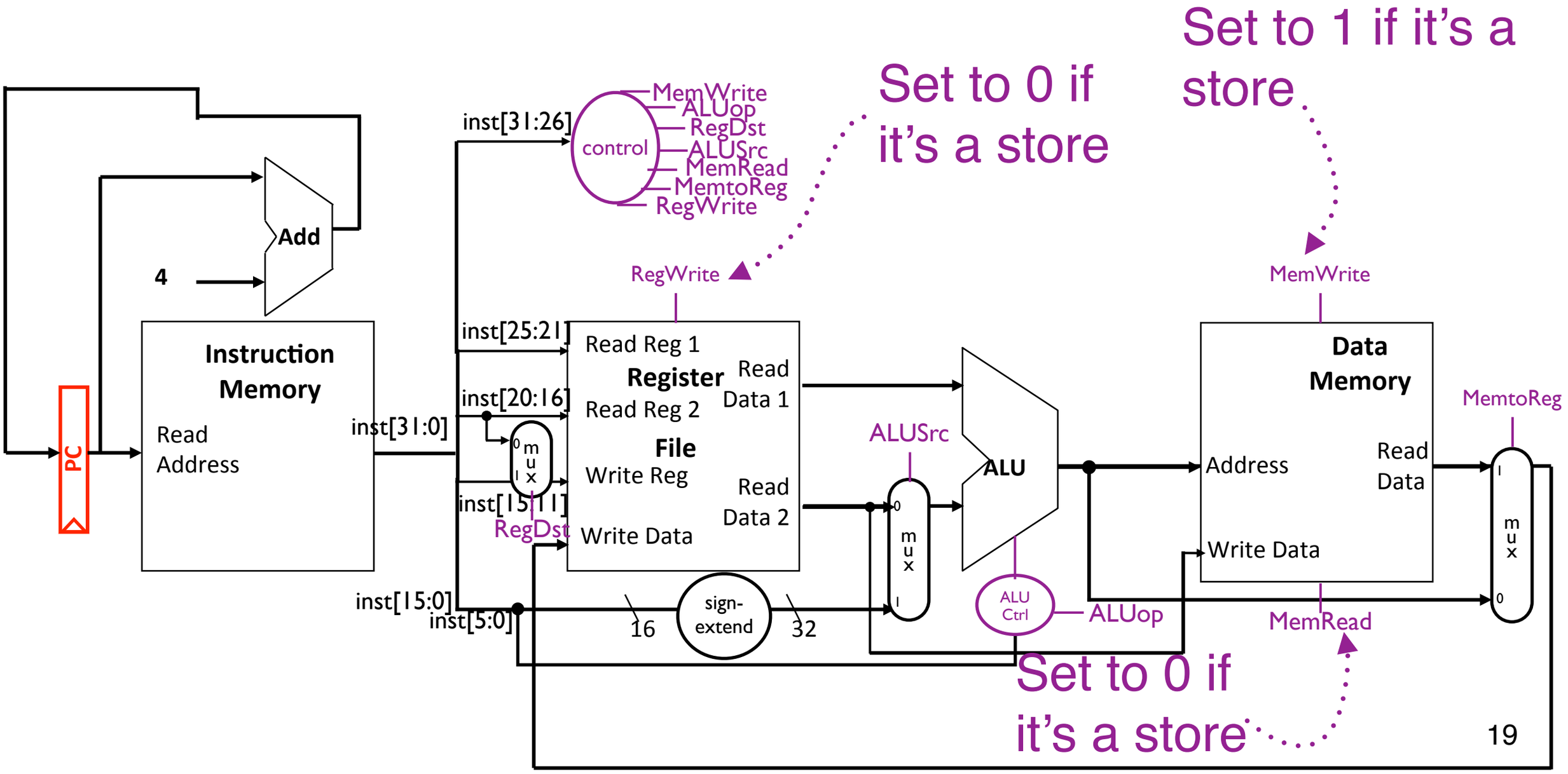
Implementing a store instruction



instruction = MEM[PC]

MEM[signext(immediate) + REG[rs]] = REG[rt]

PC = PC + 4

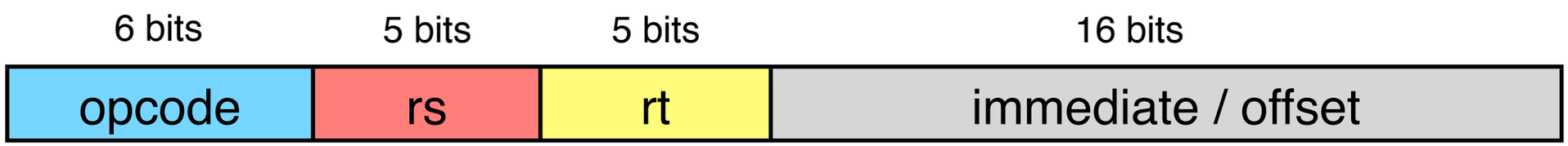


Implementing a branch instruction

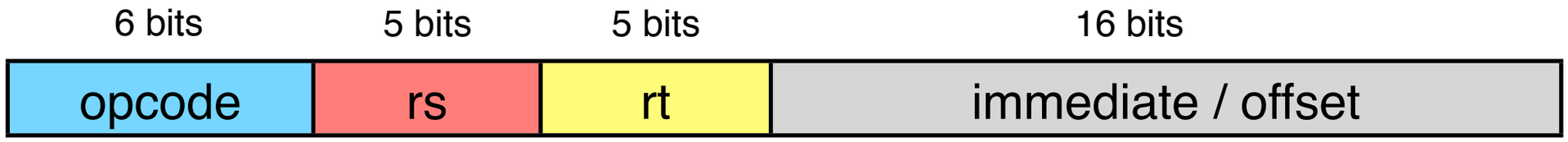
- How many of the following datapath elements is necessary for a branch instruction?
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. 1
B. 2
C. 3
D. 4
E. 5

instruction = MEM[PC]
PC = (REG[rs] == REG[rt]) ? PC + 4 + SignExtImmediate *4 : PC + 4



Implementing a branch instruction



instruction = MEM[PC]

PC = (REG[rs] == REG[rt]) ? PC + 4 + SignExtImmediate *4 : PC + 4

PCSrc = Branch & Zero

Calculate the target address

Performance of a single-cycle processor

- How many of the following statements about a single-cycle processor is correct?
  - The CPI of a single-cycle processor is always **1**
  - If the single-cycle implements lw, sw, beq, and add instructions, the sw instruction determines the cycle time
  - Hardware elements are mostly idle during a cycle
  - We can always reduce the cycle time of a single-cycle processor by supporting fewer instructions

A. 0

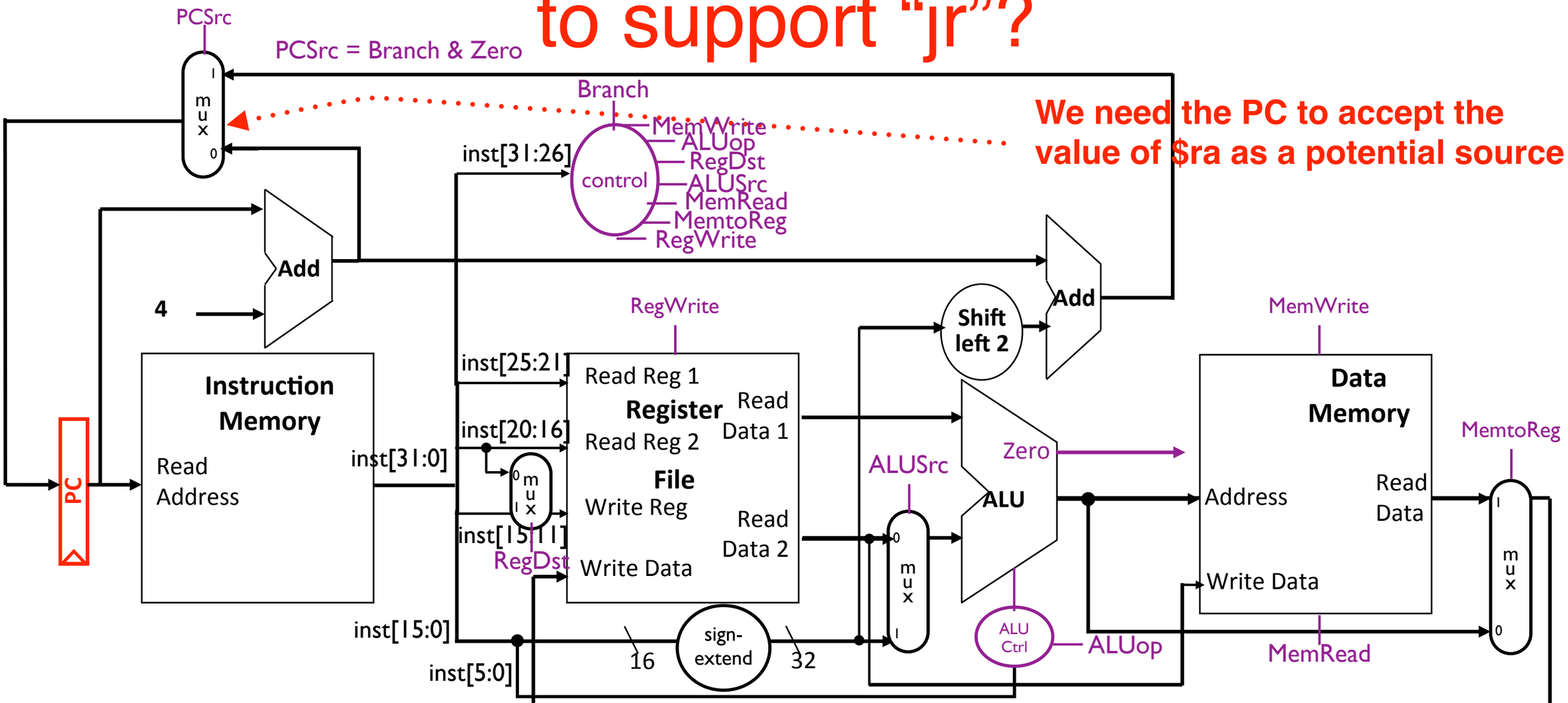
B. 1

C. 2

D. 3

E. 4

Do we need to modify the current processor to support "jr"?



A. Yes. We need both new control and datapath

B. Yes. We only need to modify datapath

C. No. But we should modify for better performance

D. No. Just changing the control signals is fine

E. Single cycle cannot do jump register

Q & A