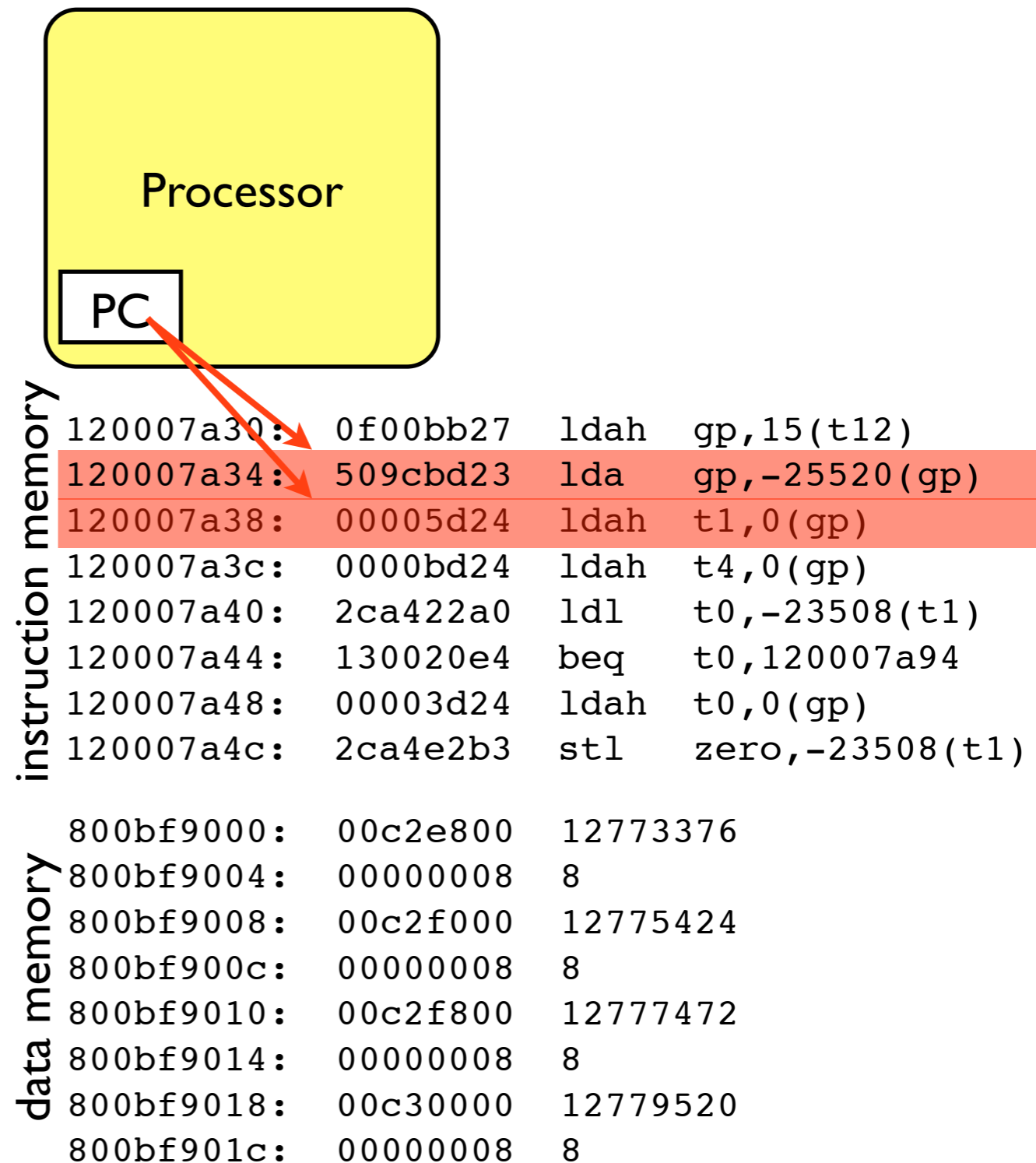


Processor Design – Single Cycle Processor

Hung-Wei Tseng

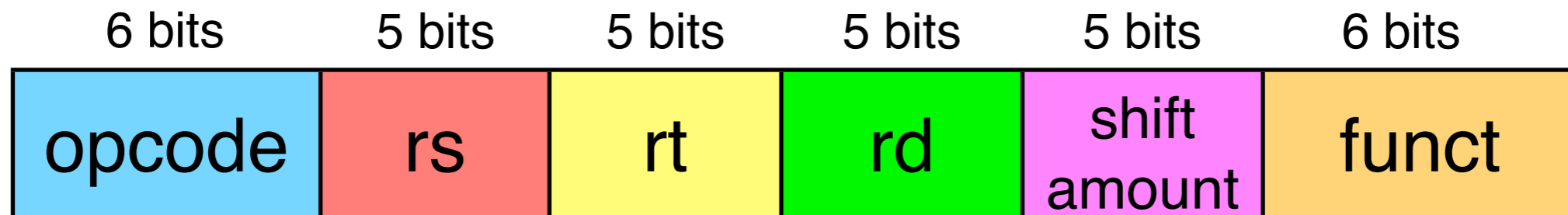
Recap: the stored-program computer

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

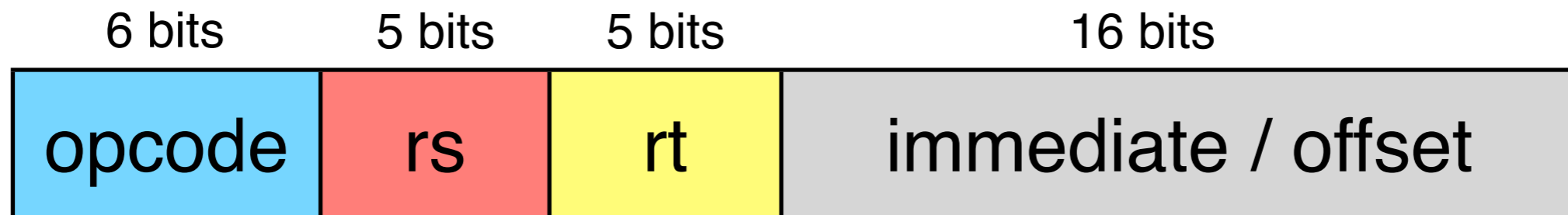


Recap: MIPS ISA

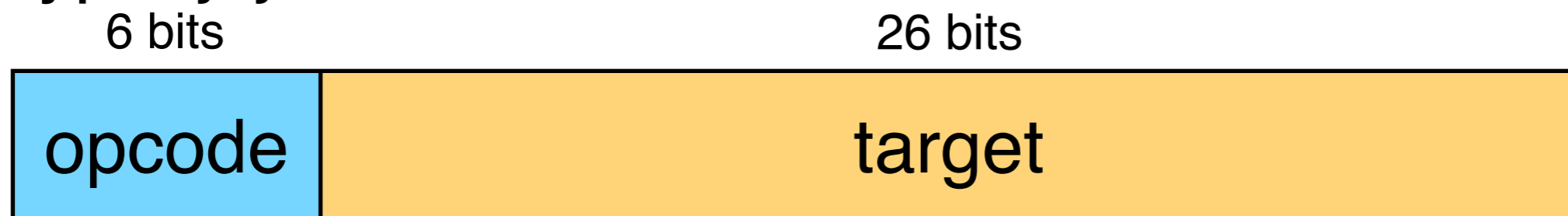
- R-type: add, sub, and etc...



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



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



Outline

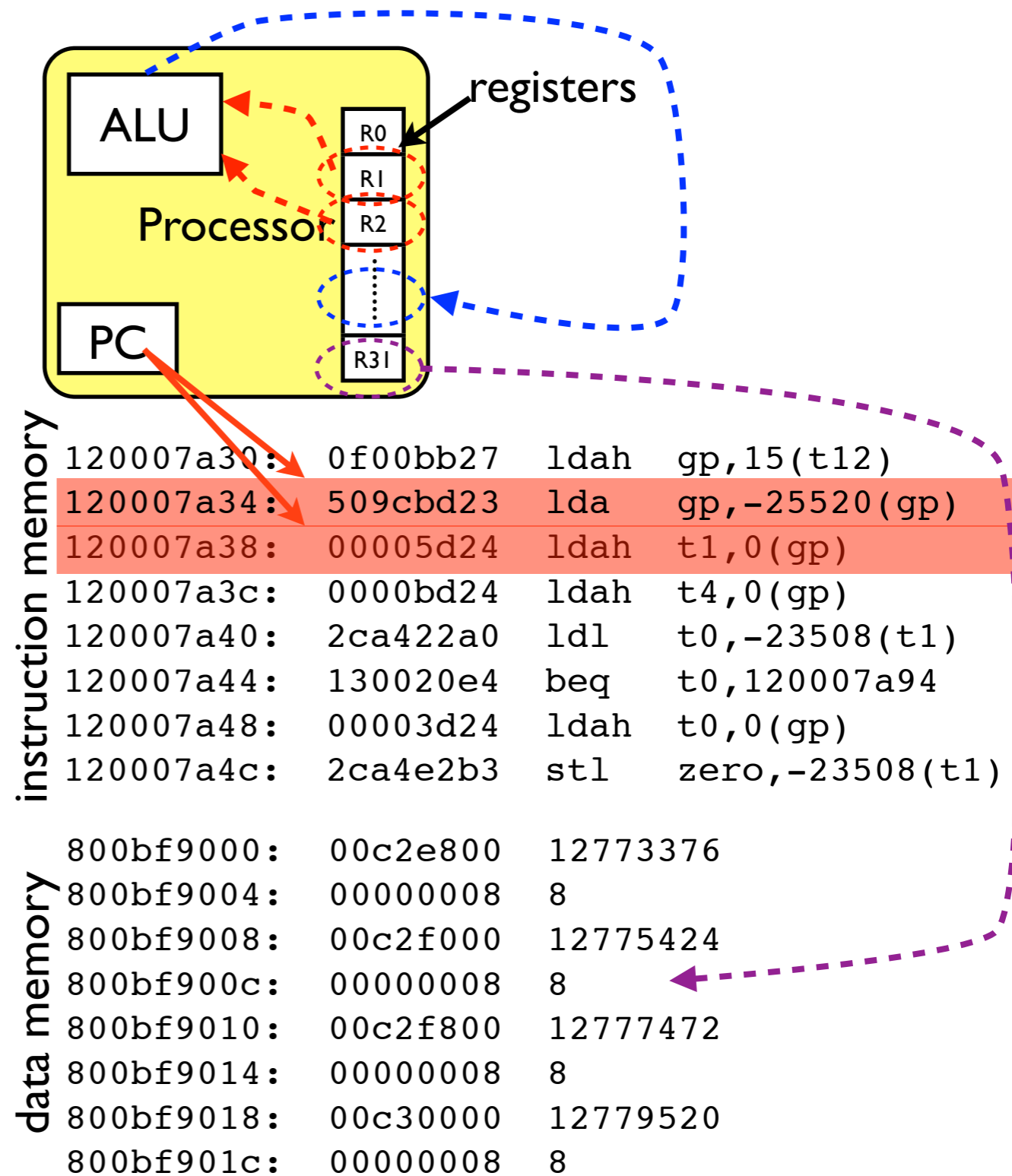
- Implementing a MIPS processor
 - Single-cycle processor
 - Pipelined processor

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

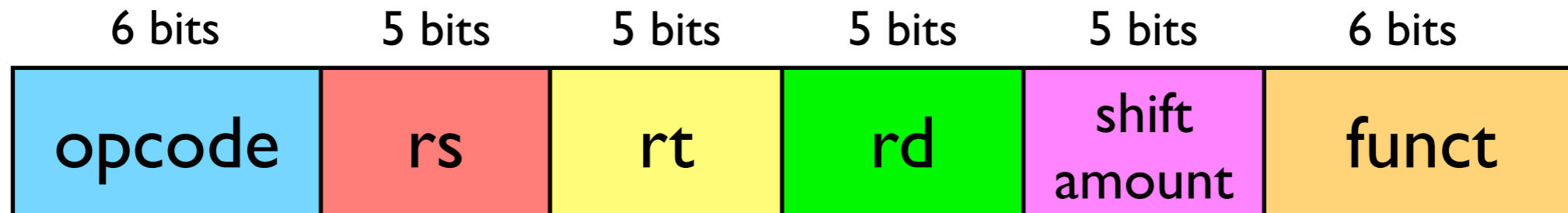
Basic steps of execution

- Instruction fetch: fetch an instruction from memory
- Decode:
 - What's the instruction?
 - Where are the operands?
- Execute
- Memory access
 - Where is my data? (The data memory address)
- Write back
 - Where to put the result
- Determine the next PC

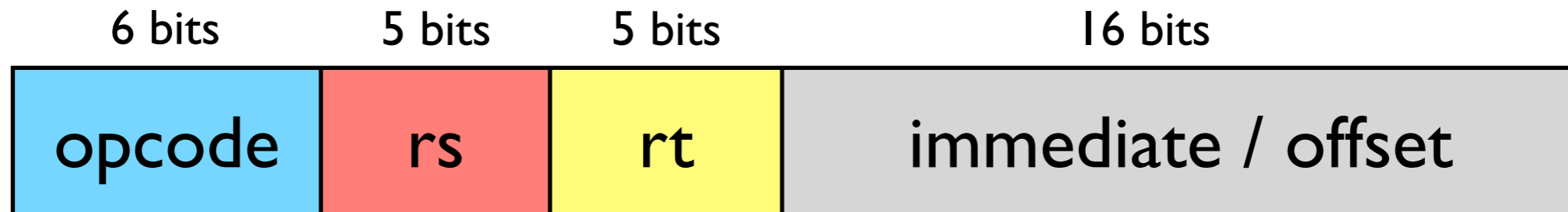


Recap: MIPS ISA

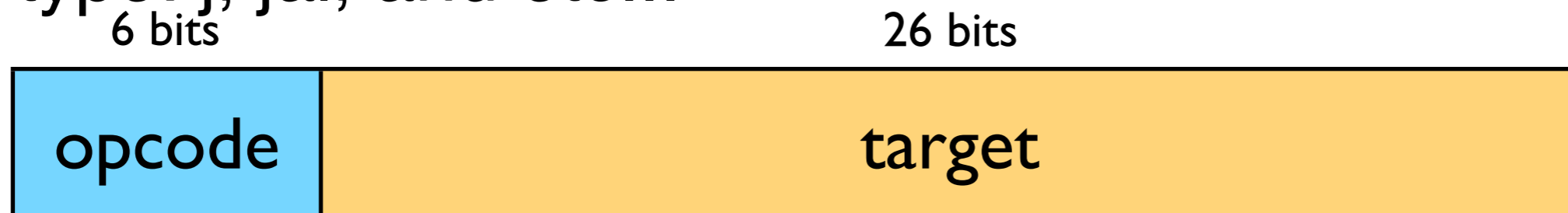
- R-type: add, sub, and etc...



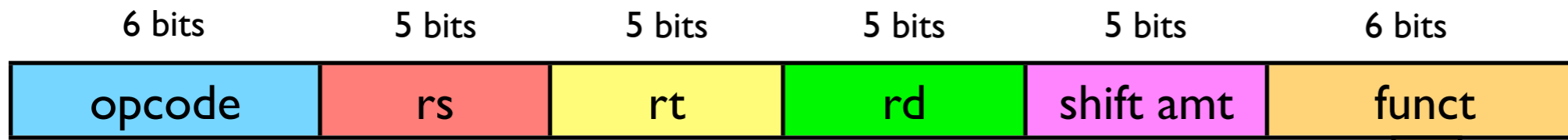
- I-type: addi, lw, sw, beq, and etc...



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

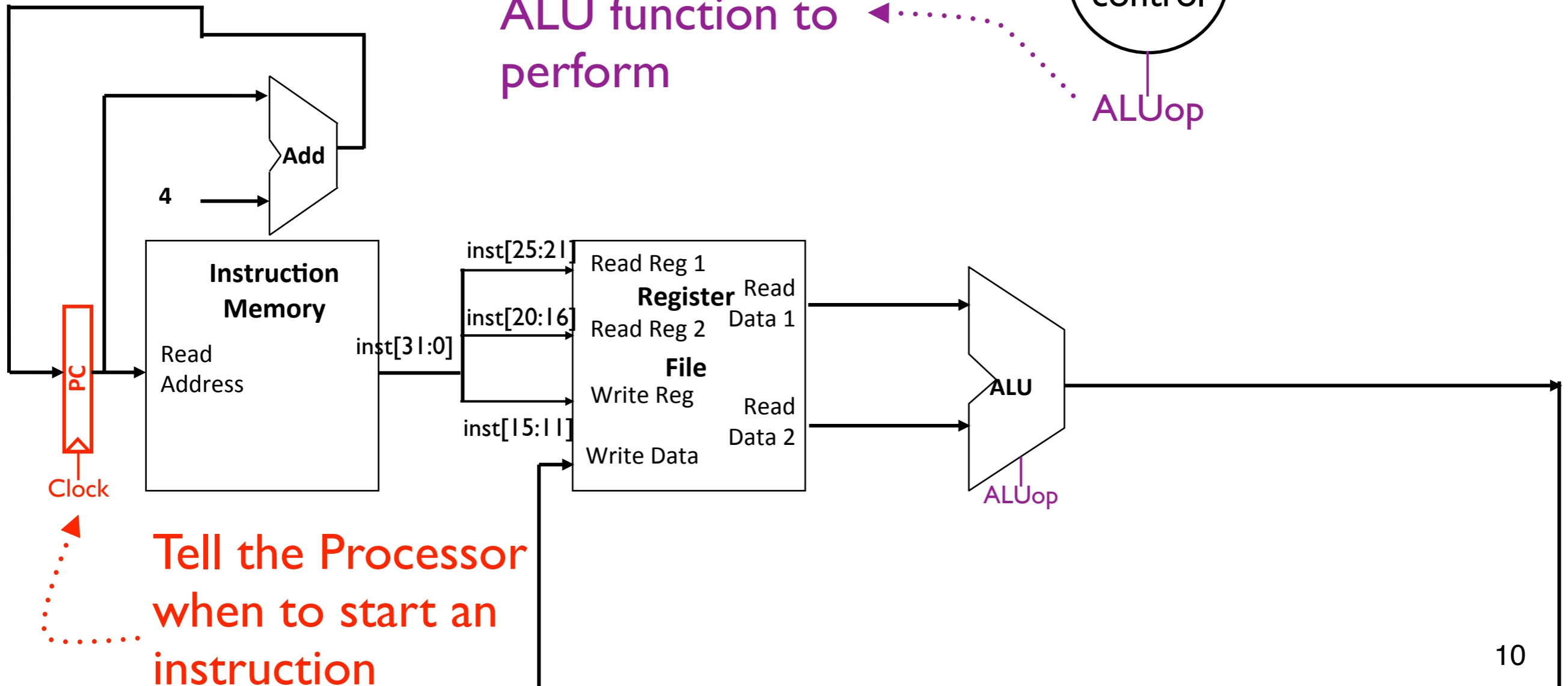
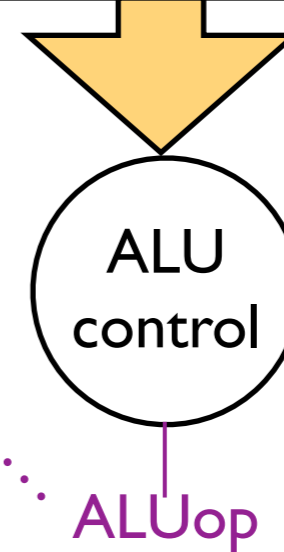


Implementing an R-type instruction

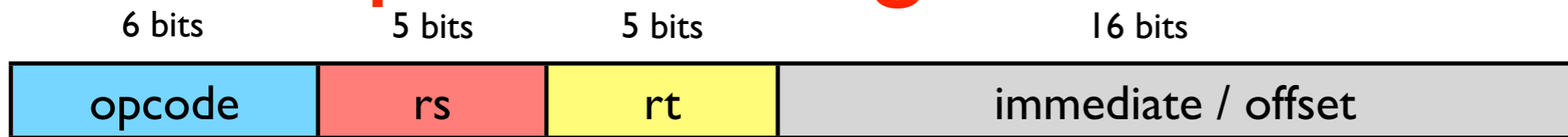


instruction = MEM[PC]
 REG[rd] = REG[rs] op REG[rt]
 PC = PC + 4

Tell the ALU what
 ALU function to
 perform



Implementing a load instruction



instruction = MEM[PC]

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

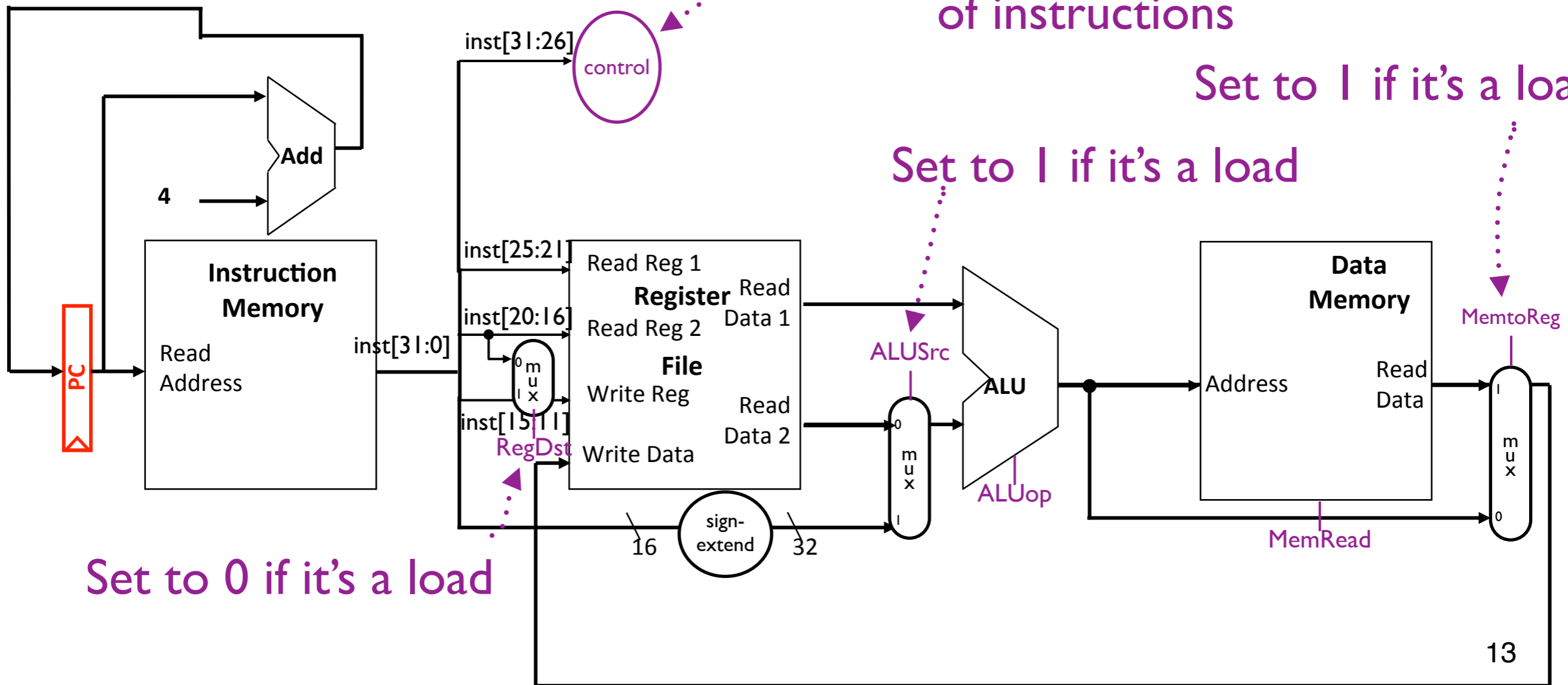
PC = PC + 4

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

6 bits

5 bits

5 bits

16 bits

opcode

rs

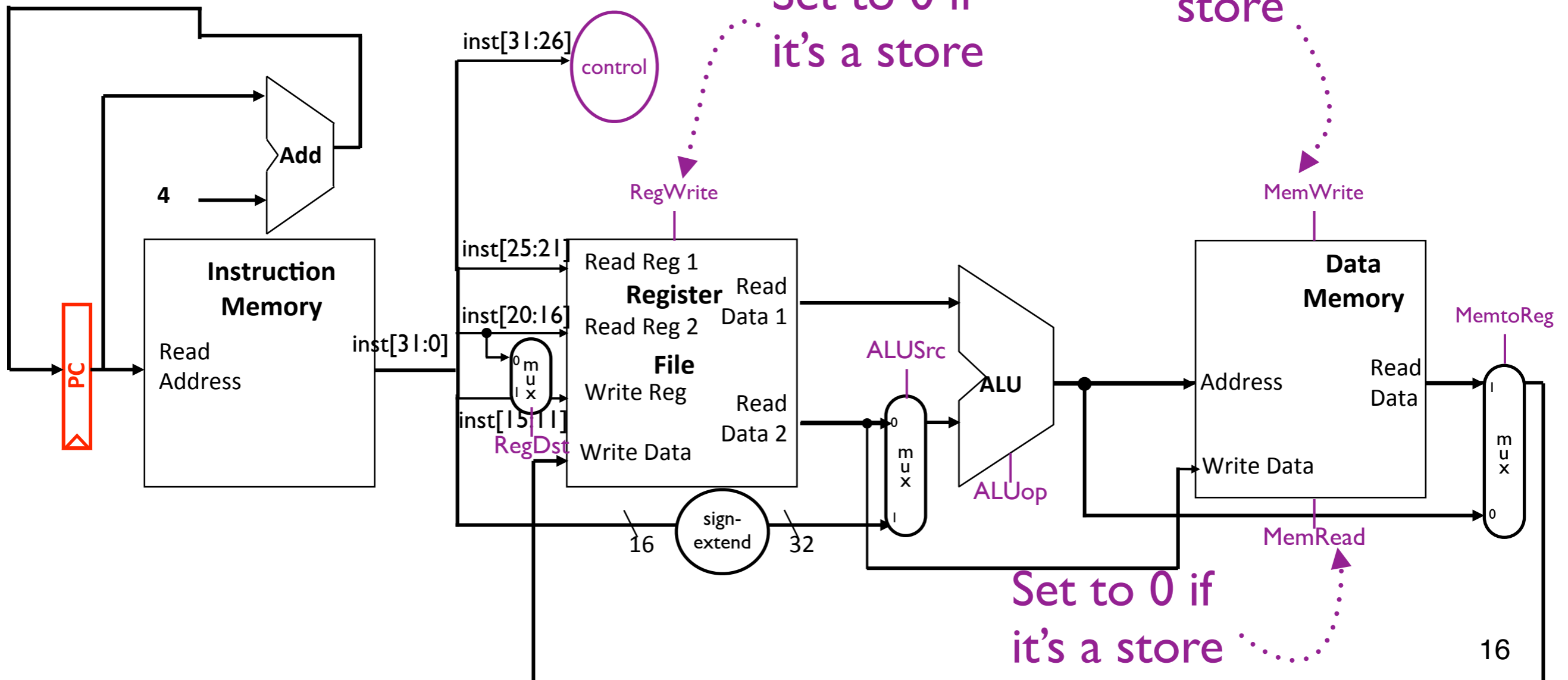
rt

immediate / offset

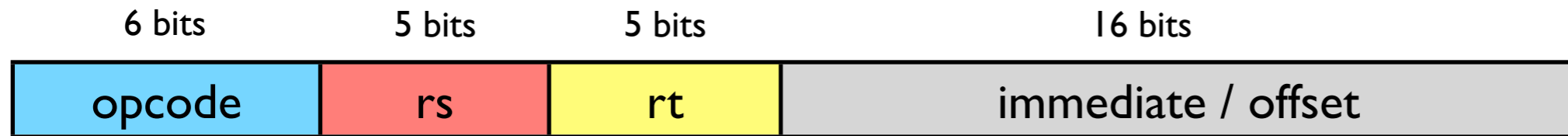
instruction = MEM[PC]

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

PC = 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

