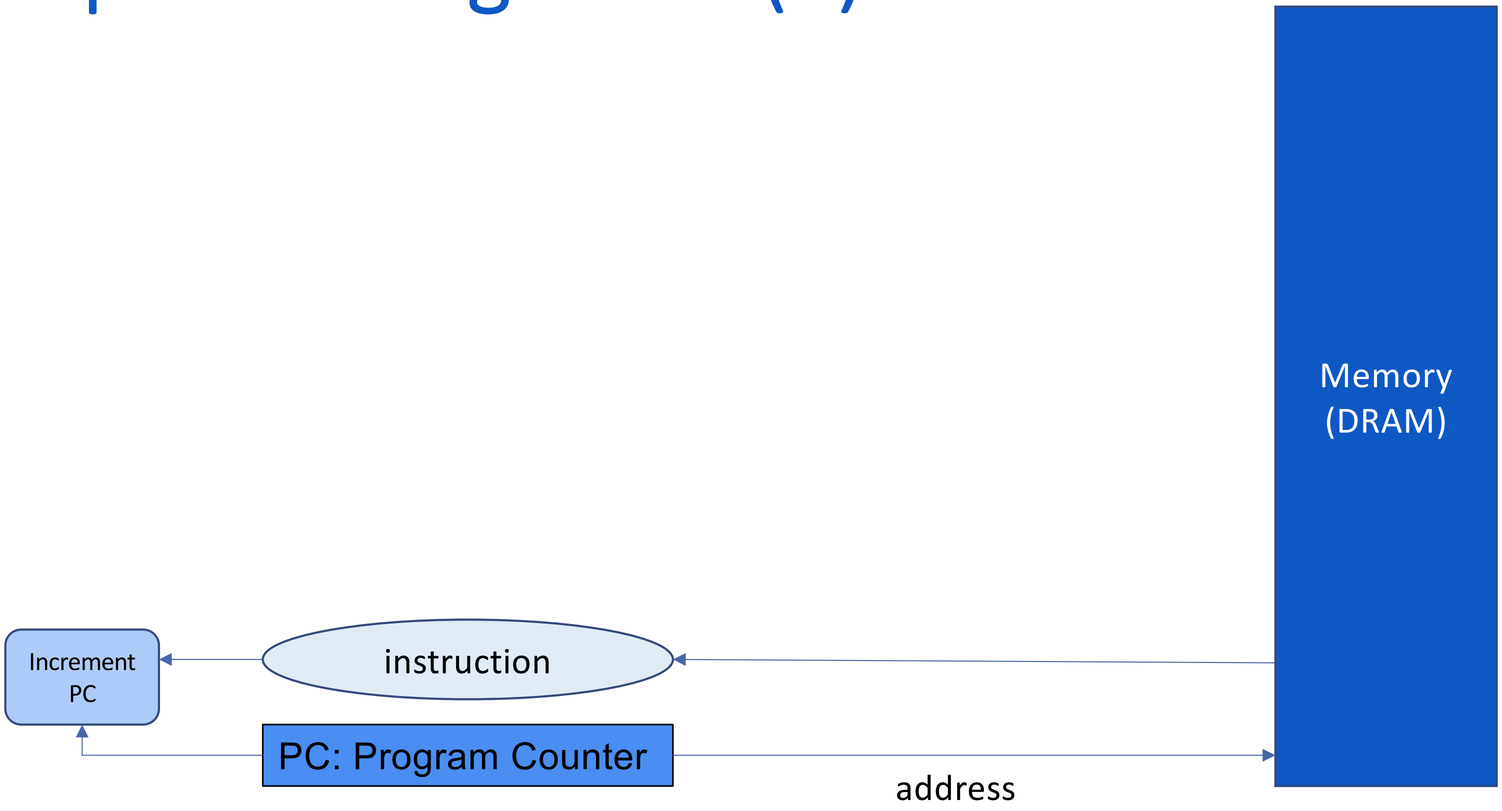
Y86 Implementation Details

- Topics
  - Detailed functionality of each execution stage
- Learning outcomes
  - Based on what must happen at each stage of execution, figure out what signals are routed where in the processor.
- Reading:
  - Section 4.3 (you can skip section 4.3.4)

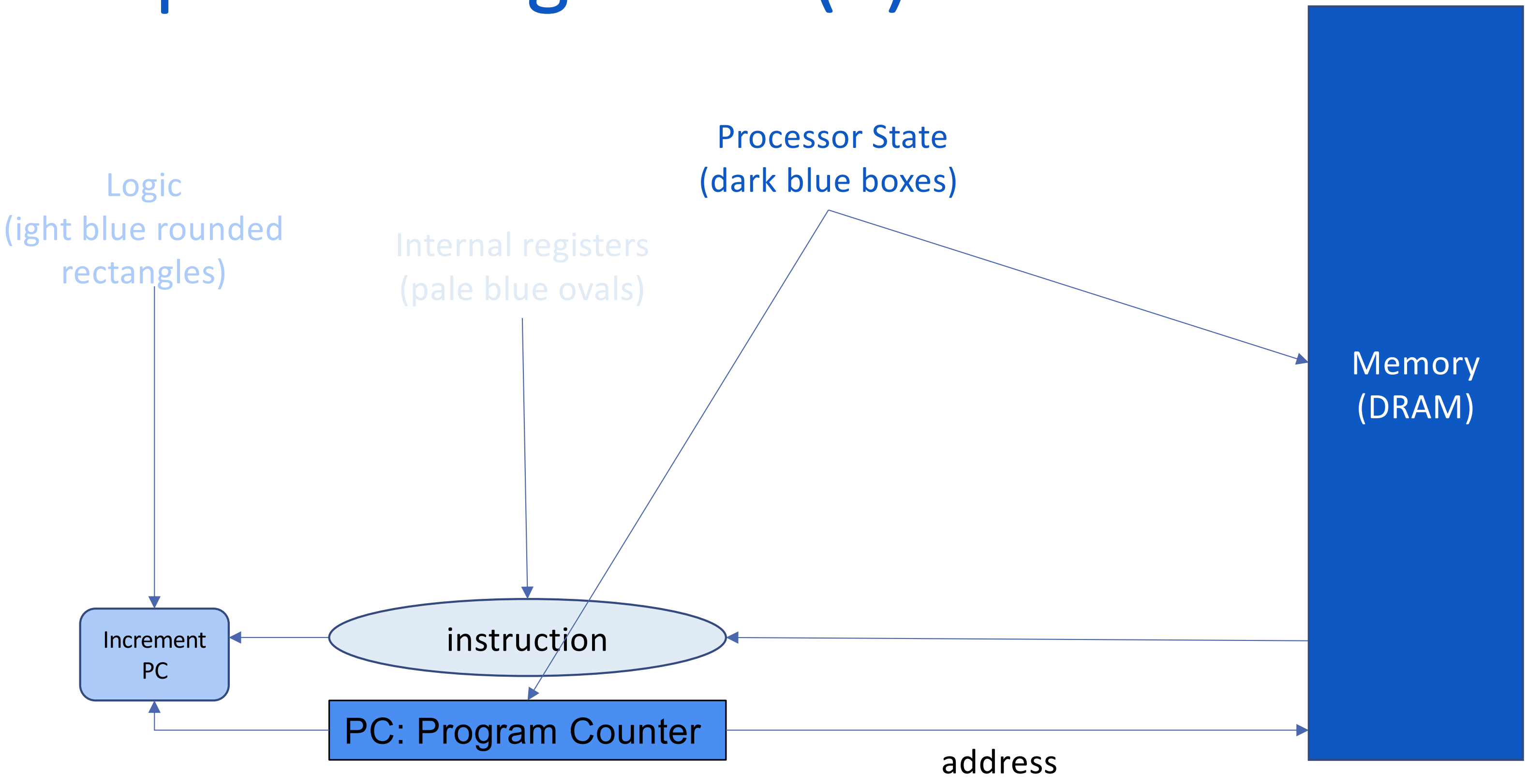
FETCH

- What must we do in this stage:
  - Read instruction from memory
    - How large is the instruction?
  - Break the instruction apart into various pieces.
- What parts of the processor are involved:
  - The program counter
  - Memory
  - **PC-increment**
  - **Logic to identify invalid instruction**

Implementing Fetch (1)



Implementing Fetch (1)

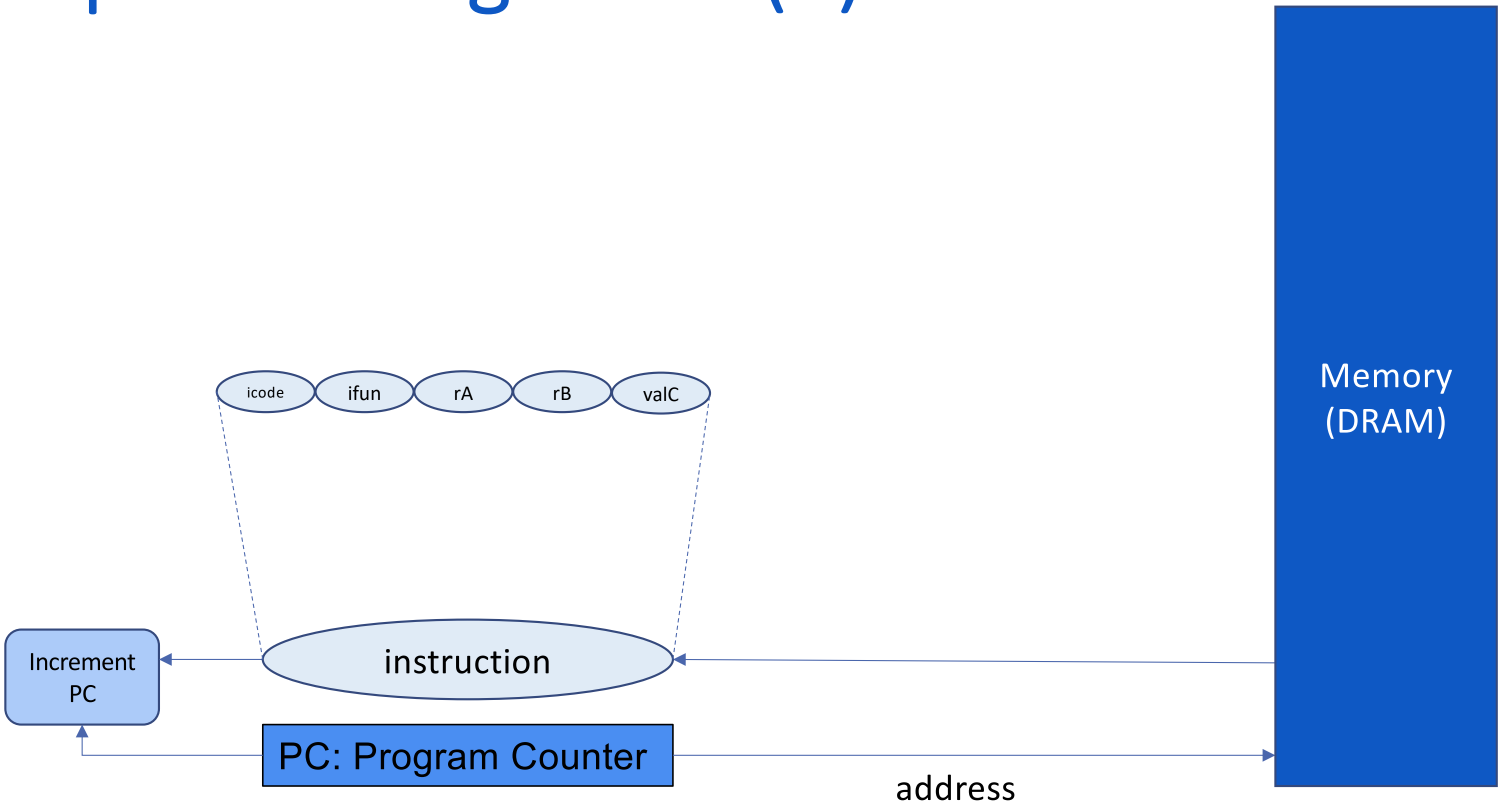


Logic
(ight blue rounded
rectangles)

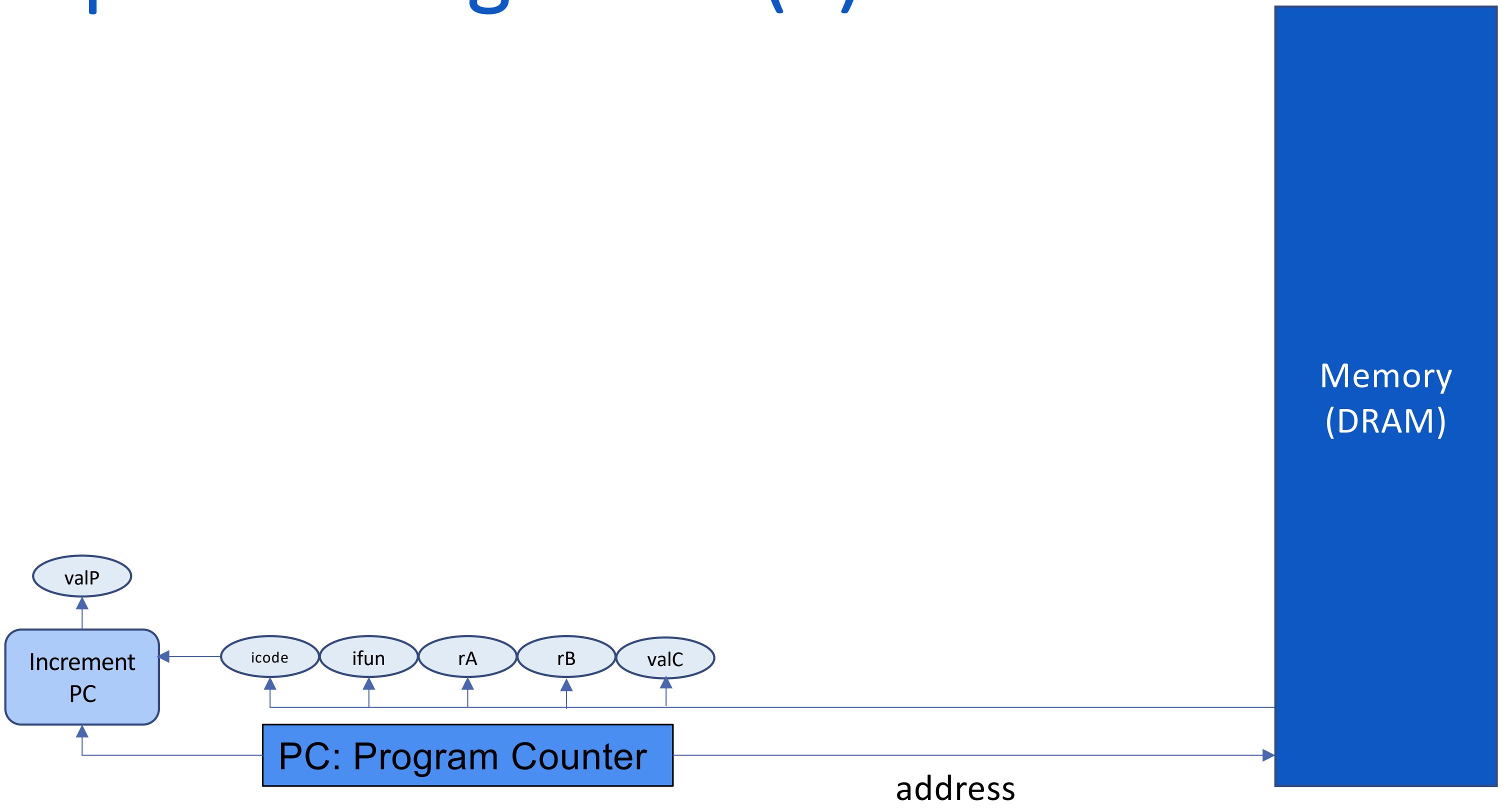
Internal registers
(pale blue ovals)

Processor State
(dark blue boxes)

Implementing Fetch (2)



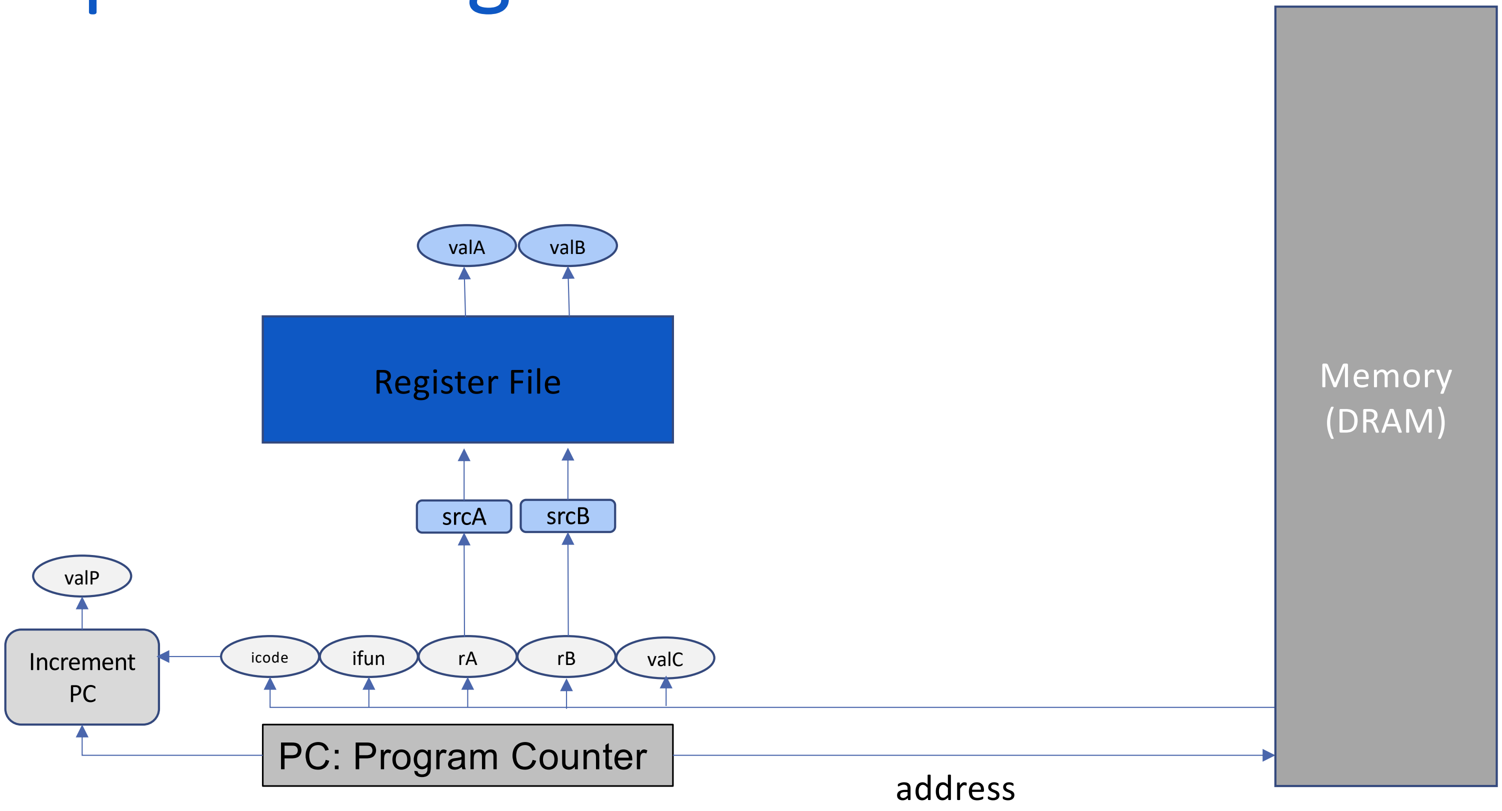
Implementing Fetch (3)



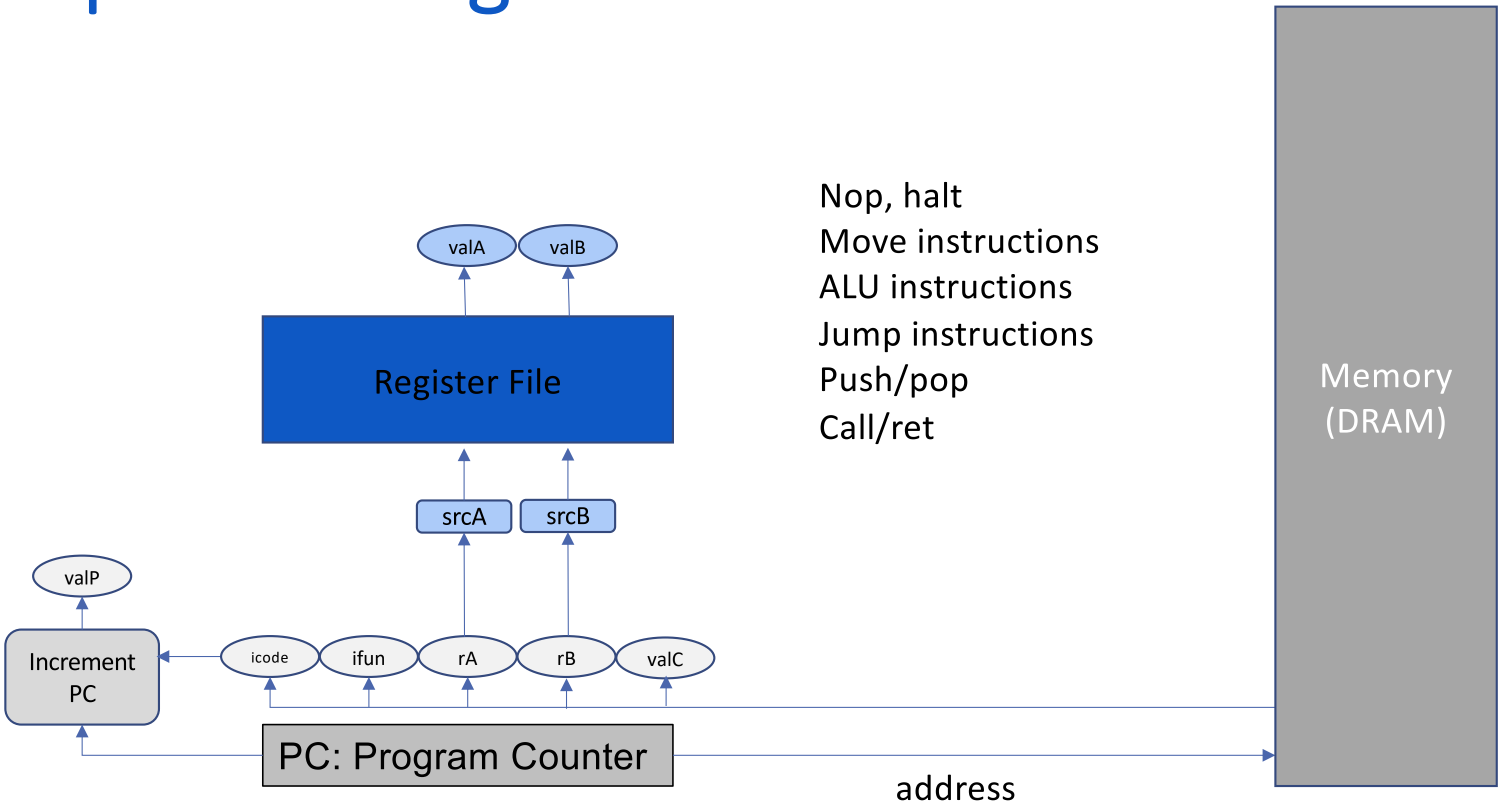
Decode

- What must we do in this stage:
  - Determine what to read from the register file
  - Read those values
- What parts of the processor are involved:
  - rA, rB from the Instruction
  - Register file
  - Logic to determine which registers are used to produce valA and valB

Implementing Decode

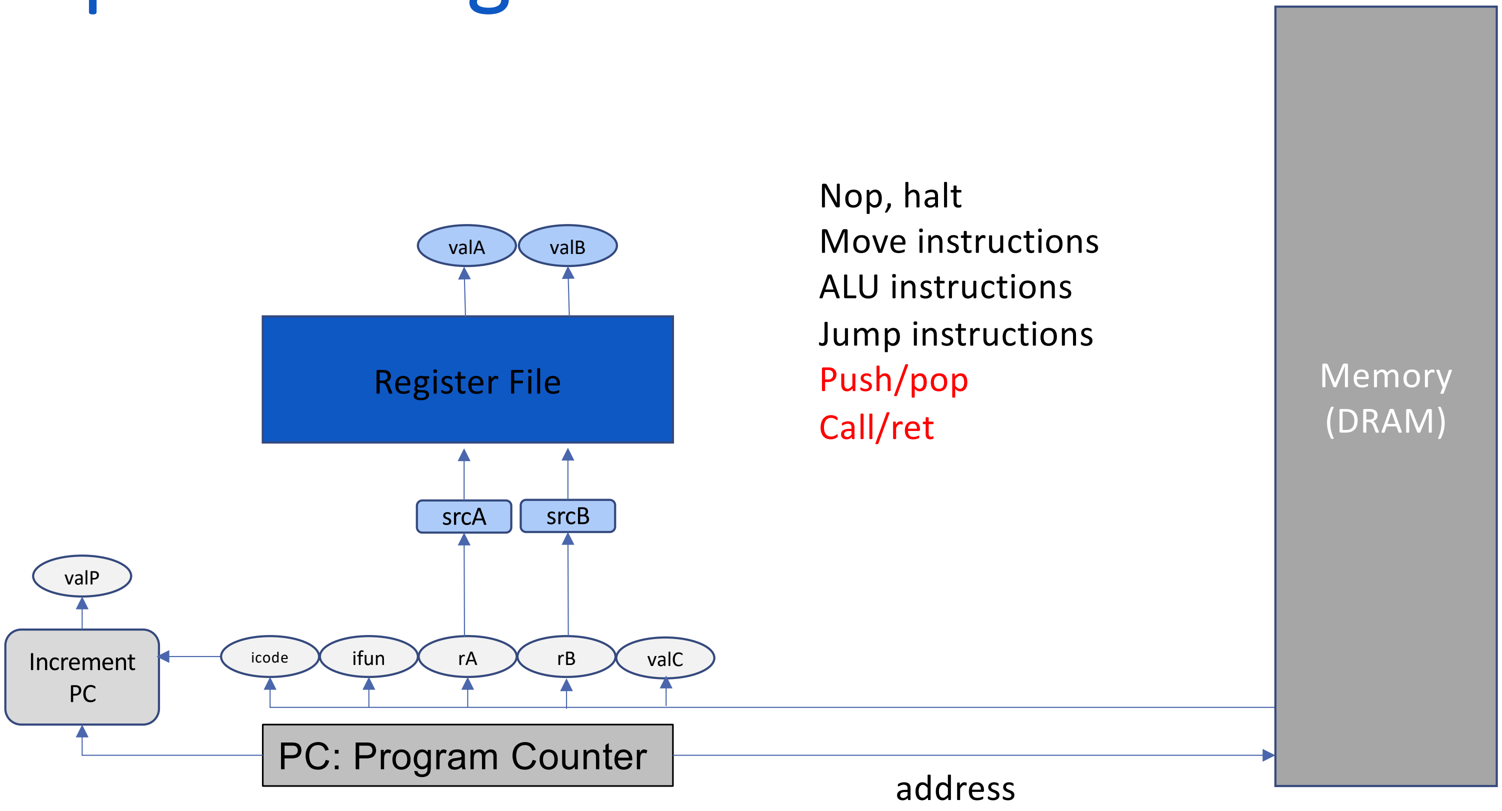


Implementing Decode



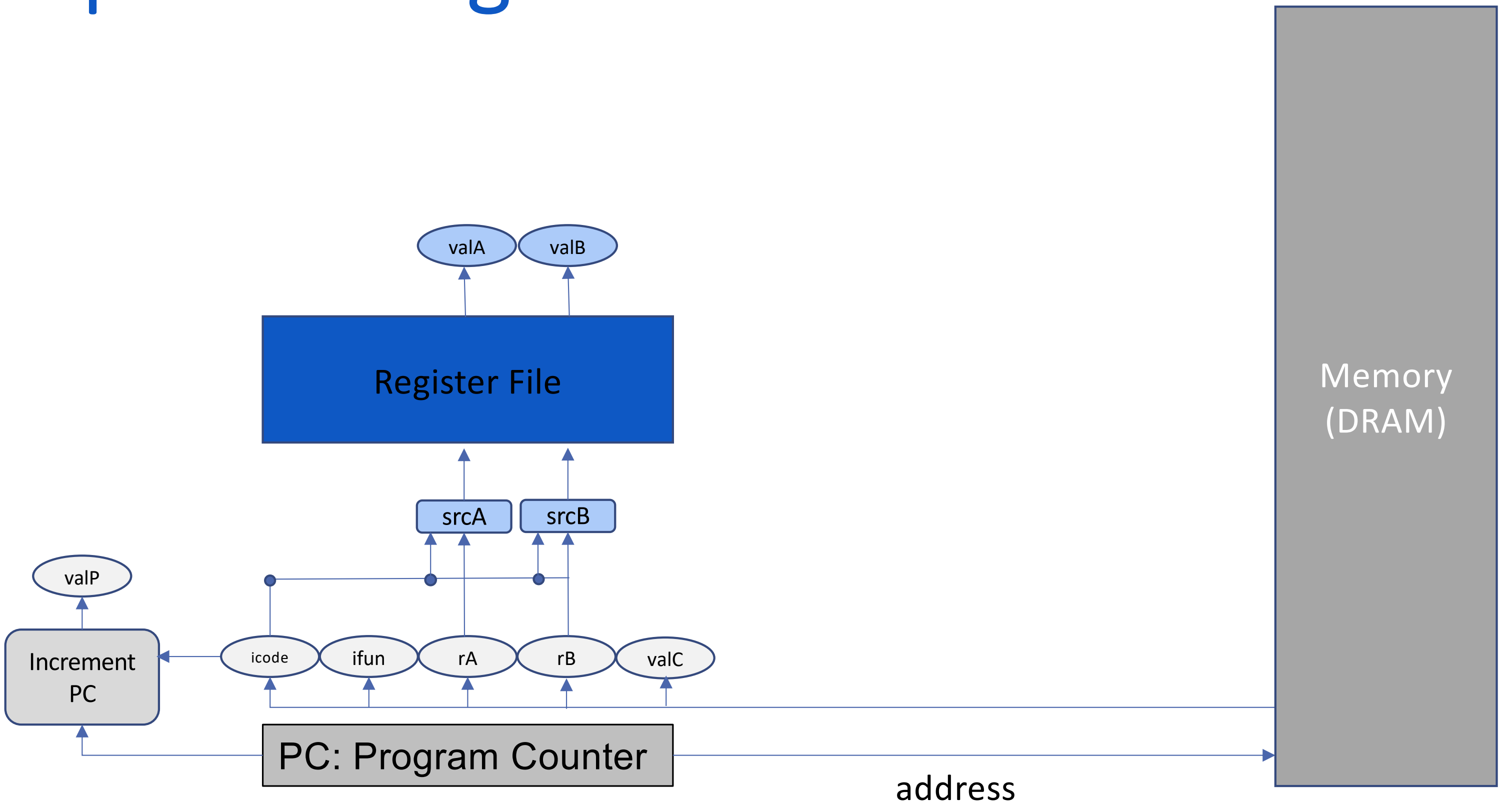
Nop, halt
Move instructions
ALU instructions
Jump instructions
Push/pop
Call/ret

Implementing Decode



Nop, halt
Move instructions
ALU instructions
Jump instructions
Push/pop
Call/ret

Implementing Decode



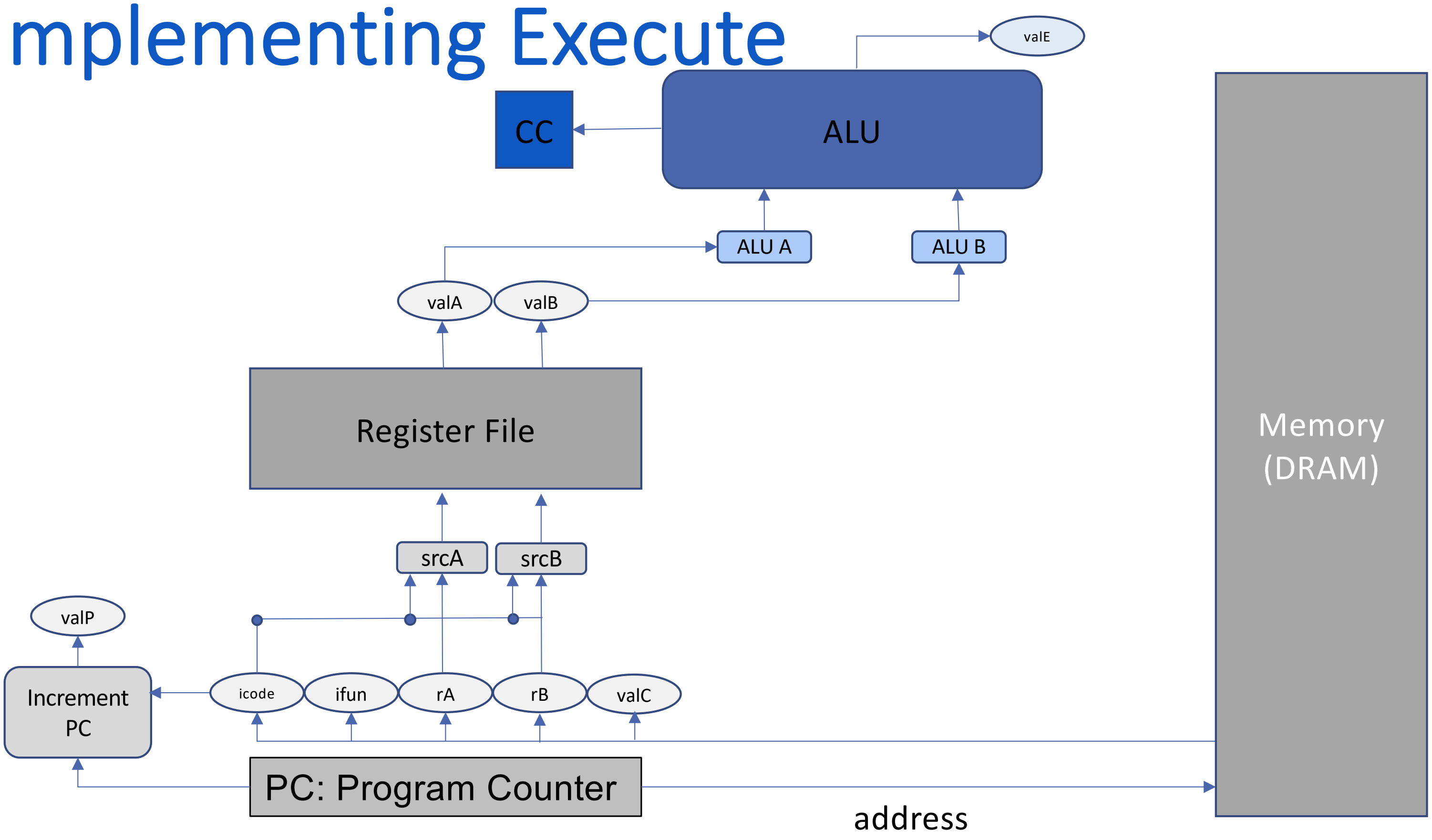
Decode

- What must we do in this stage:
  - Determine what to read from the register file
  - Read those values
- What parts of the processor are involved:
  - rA, rB and icode from the Instruction
  - Register file
  - Logic to determine which registers are used to produce valA and valB

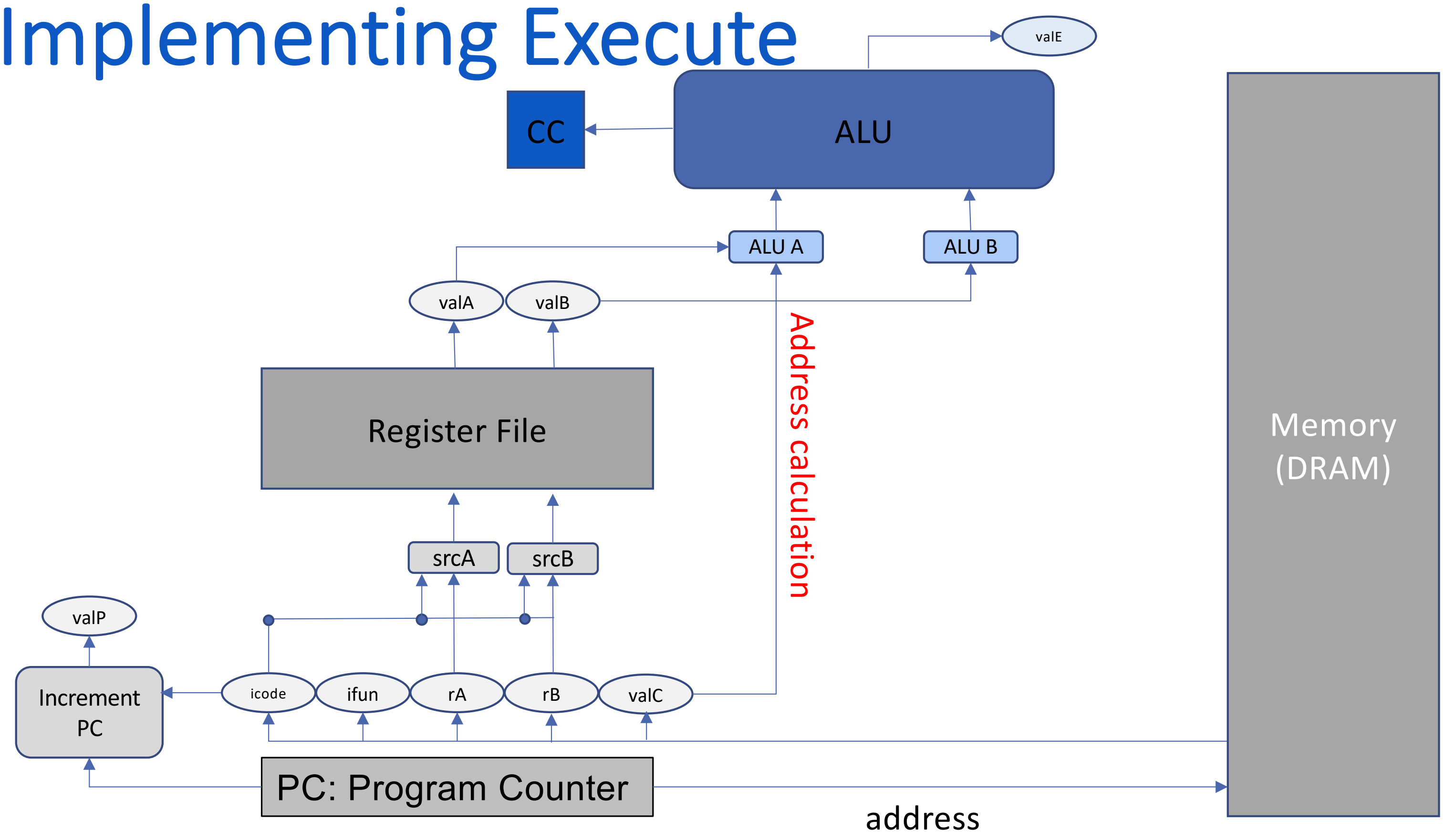
Execute

- What must we do in this stage:
  - Use the ALU
  - Set condition codes
- What parts of the processor are involved:
  - valA, valB from the register file
  - valC from the instruction
  - ALU
  - Condition codes
  - Logic to a) select inputs to ALU, b) set condition codes

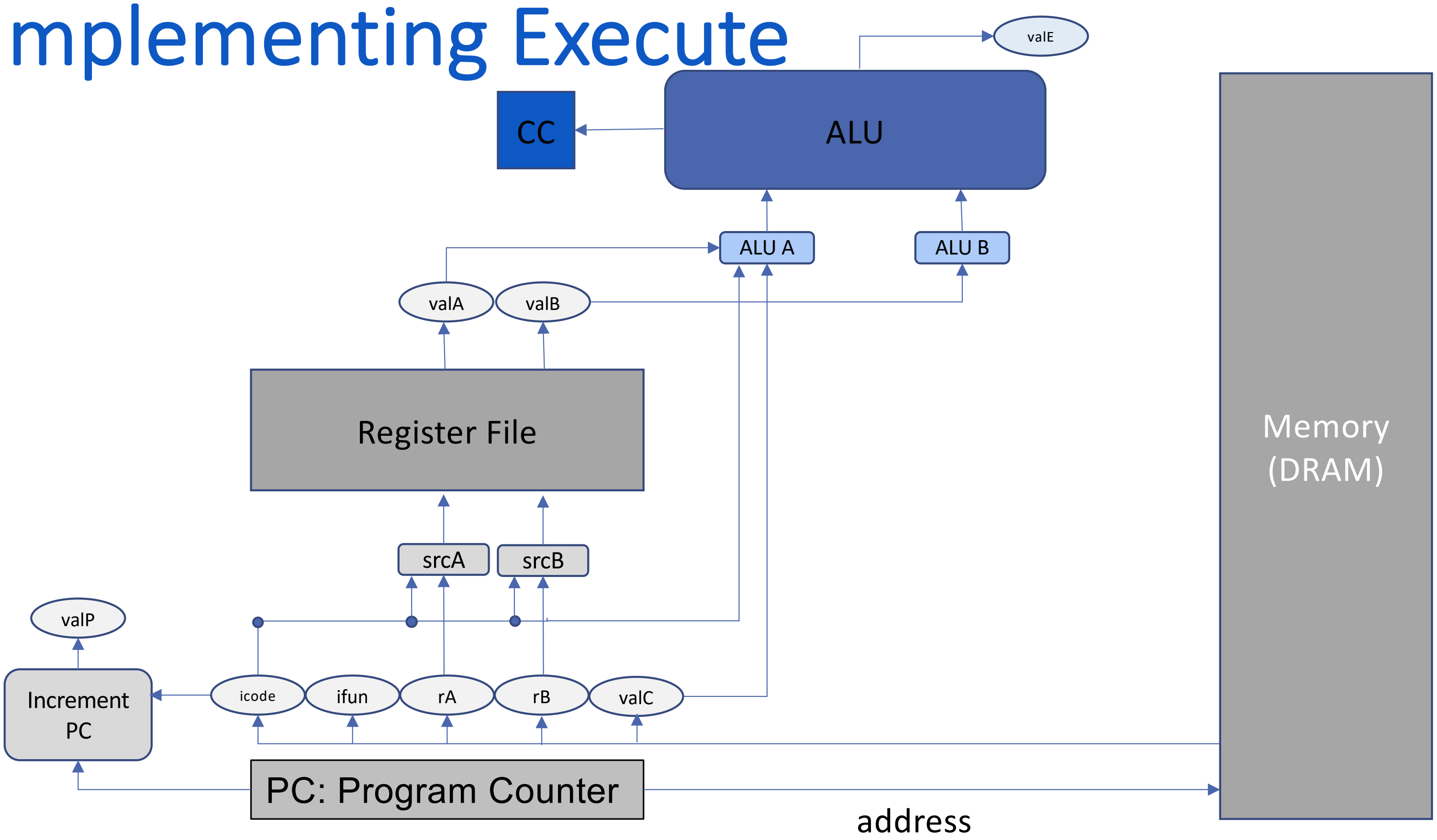
Implementing Execute



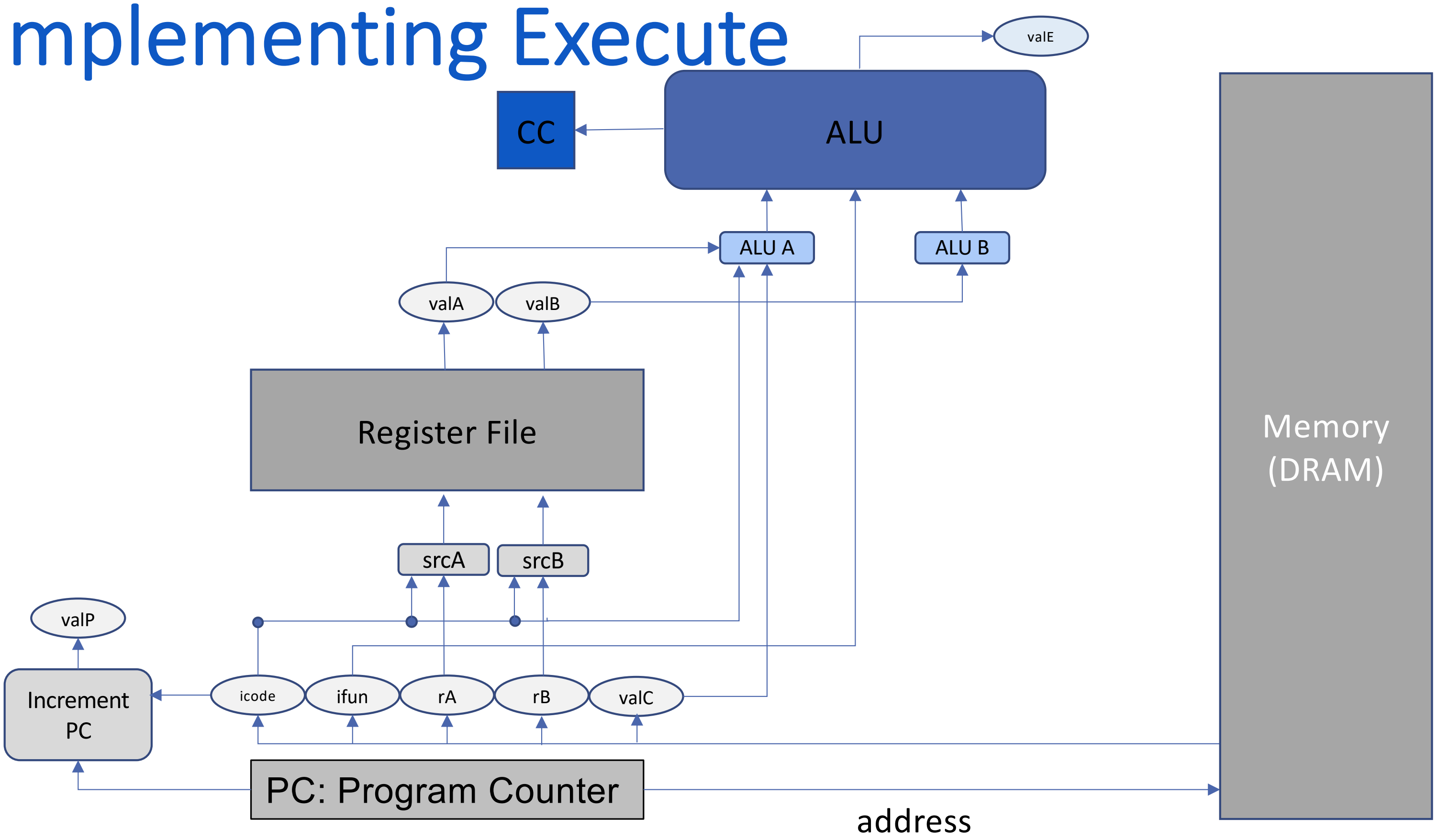
Implementing Execute



Implementing Execute

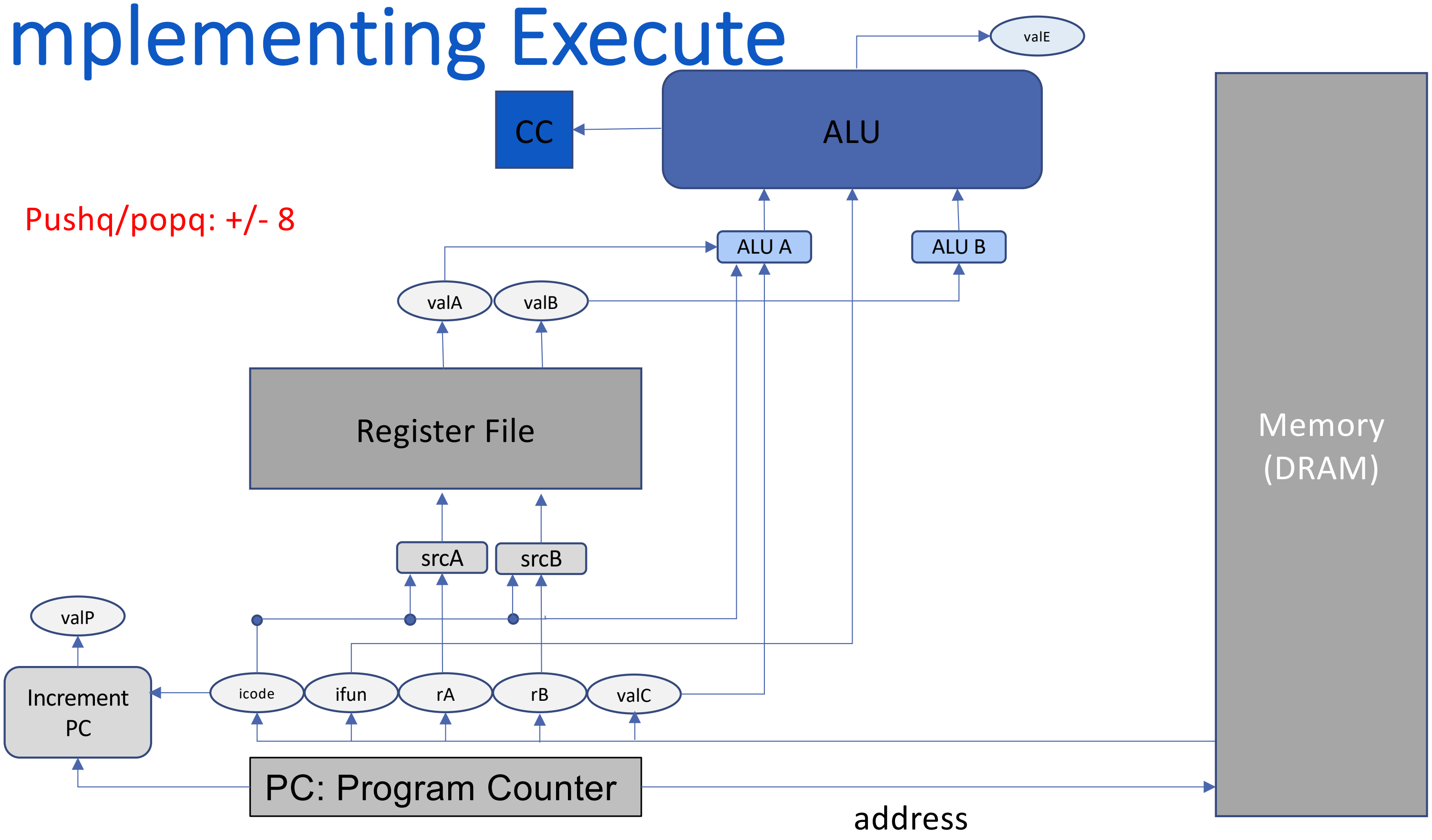


Implementing Execute

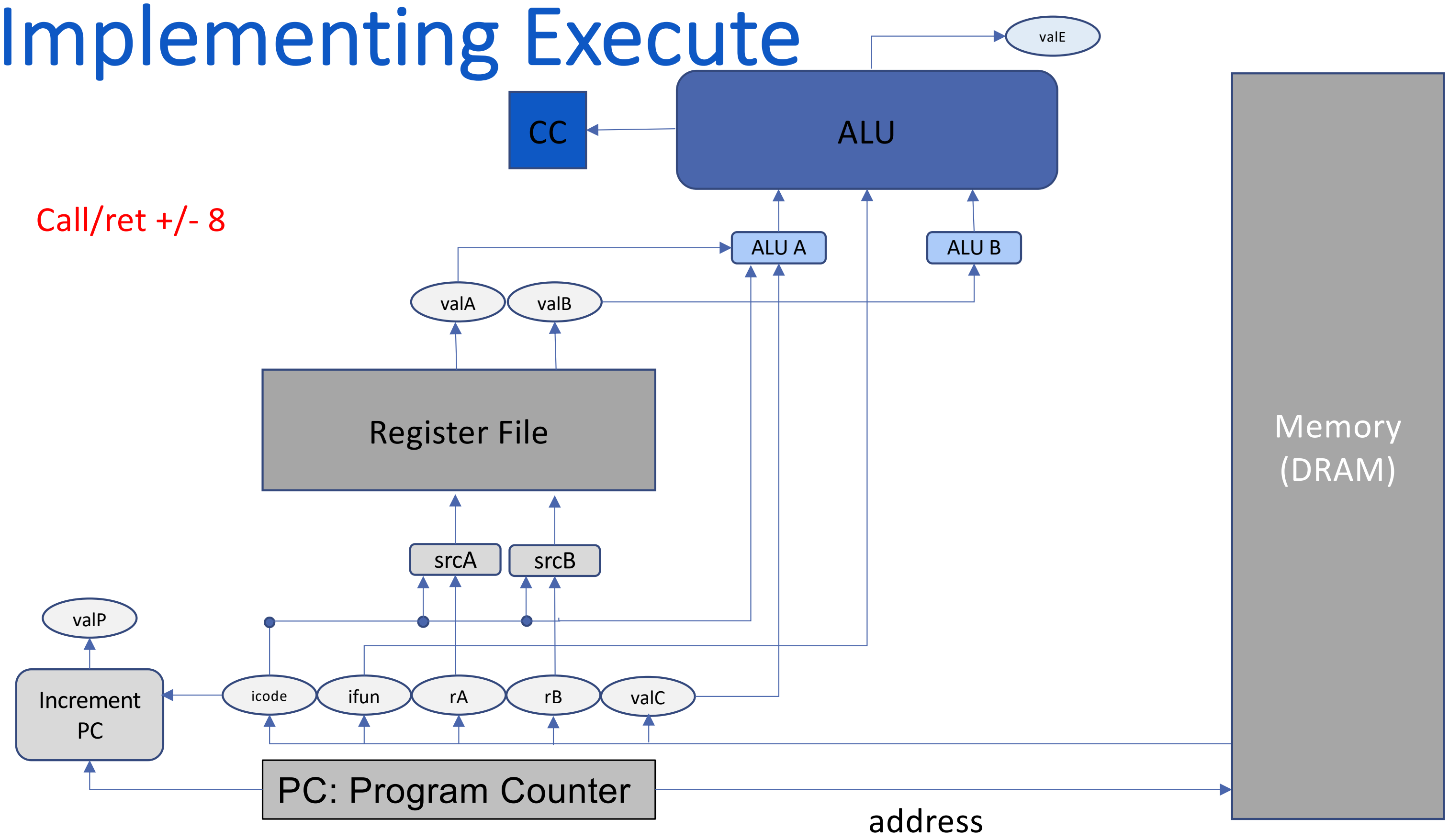


Implementing Execute

Pushq/popq: +/- 8



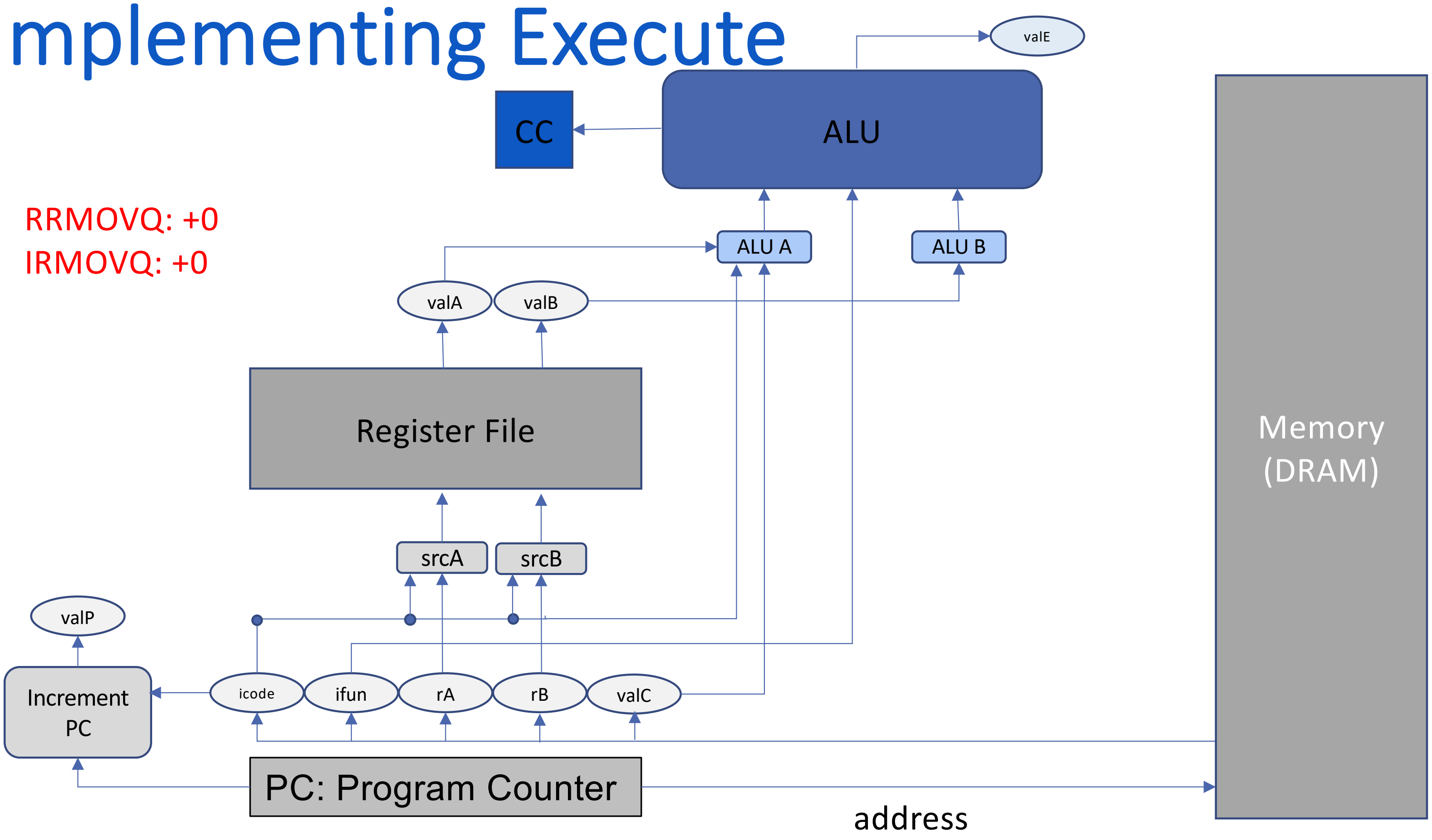
Implementing Execute



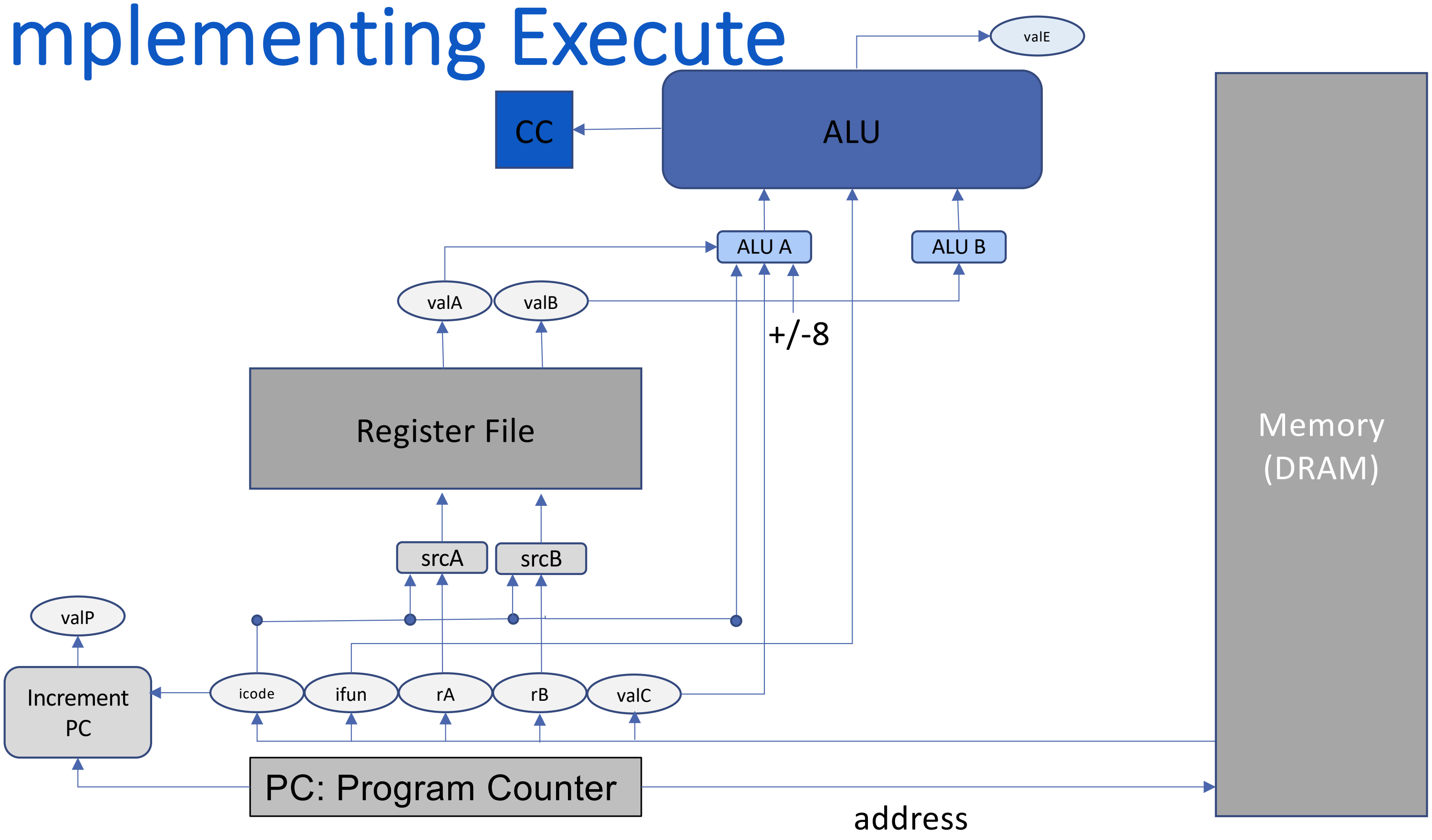
Call/ret +/- 8

Implementing Execute

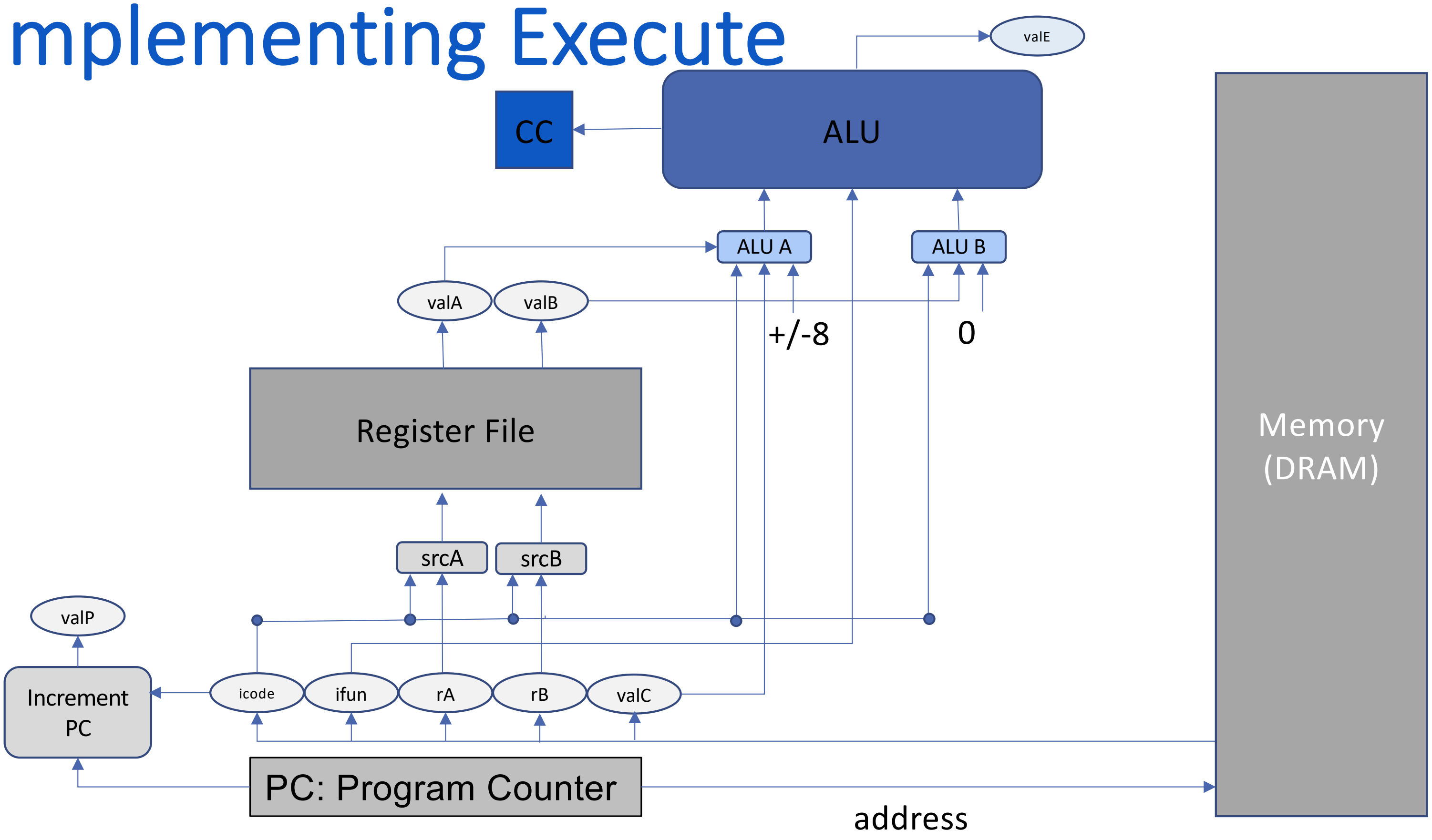
RRMOVQ: +0
IRMOVQ: +0



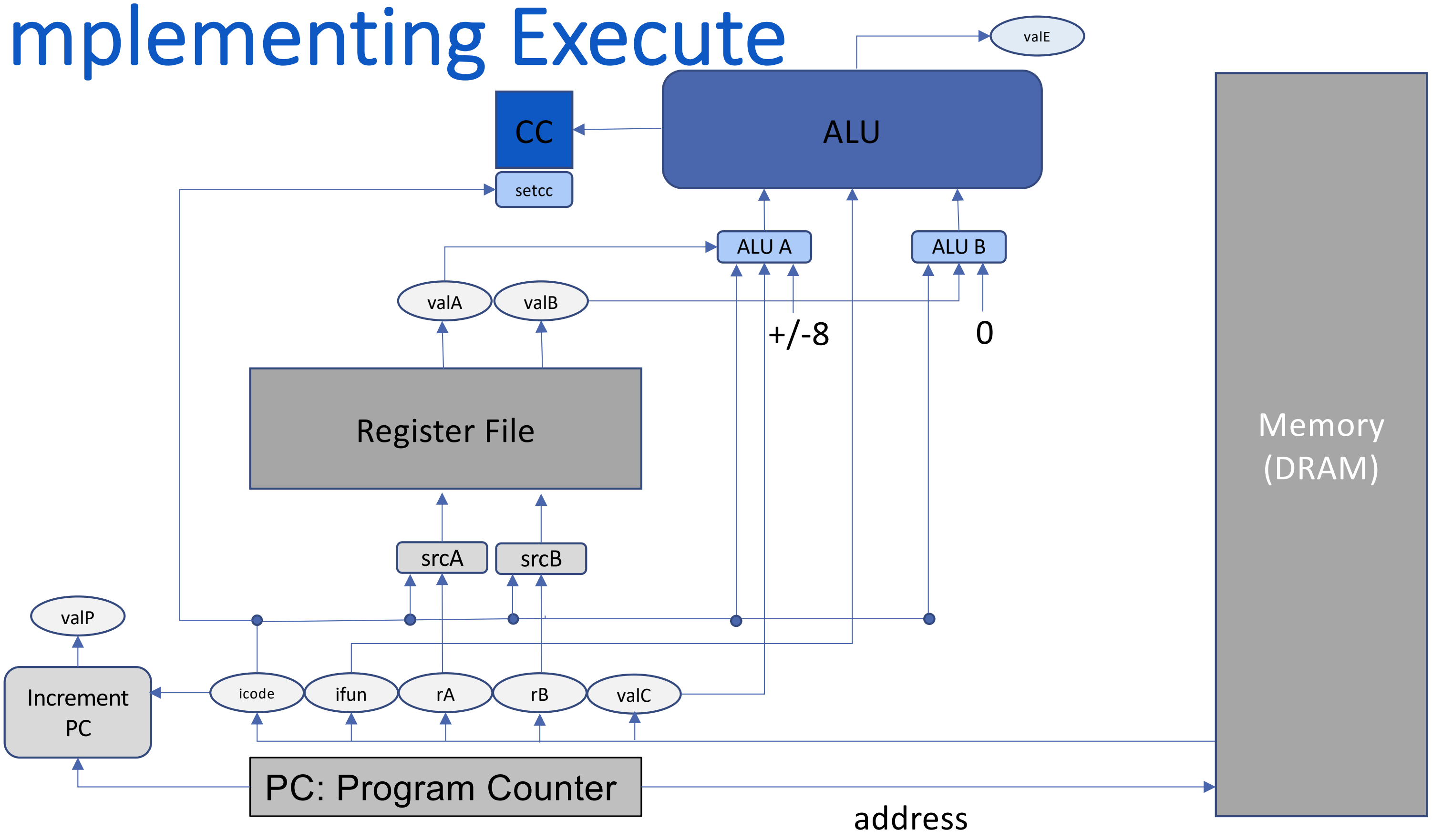
Implementing Execute



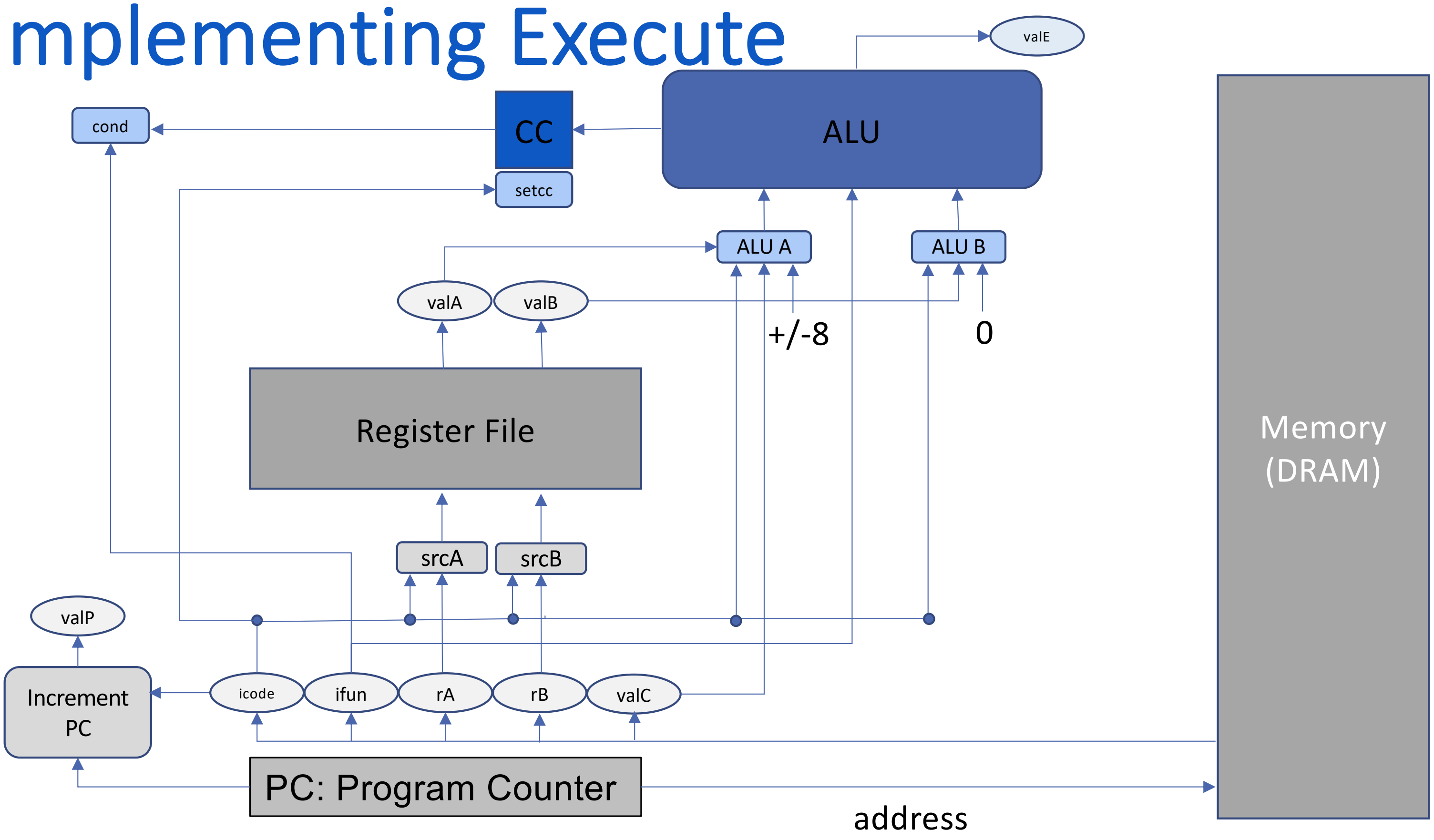
Implementing Execute



Implementing Execute



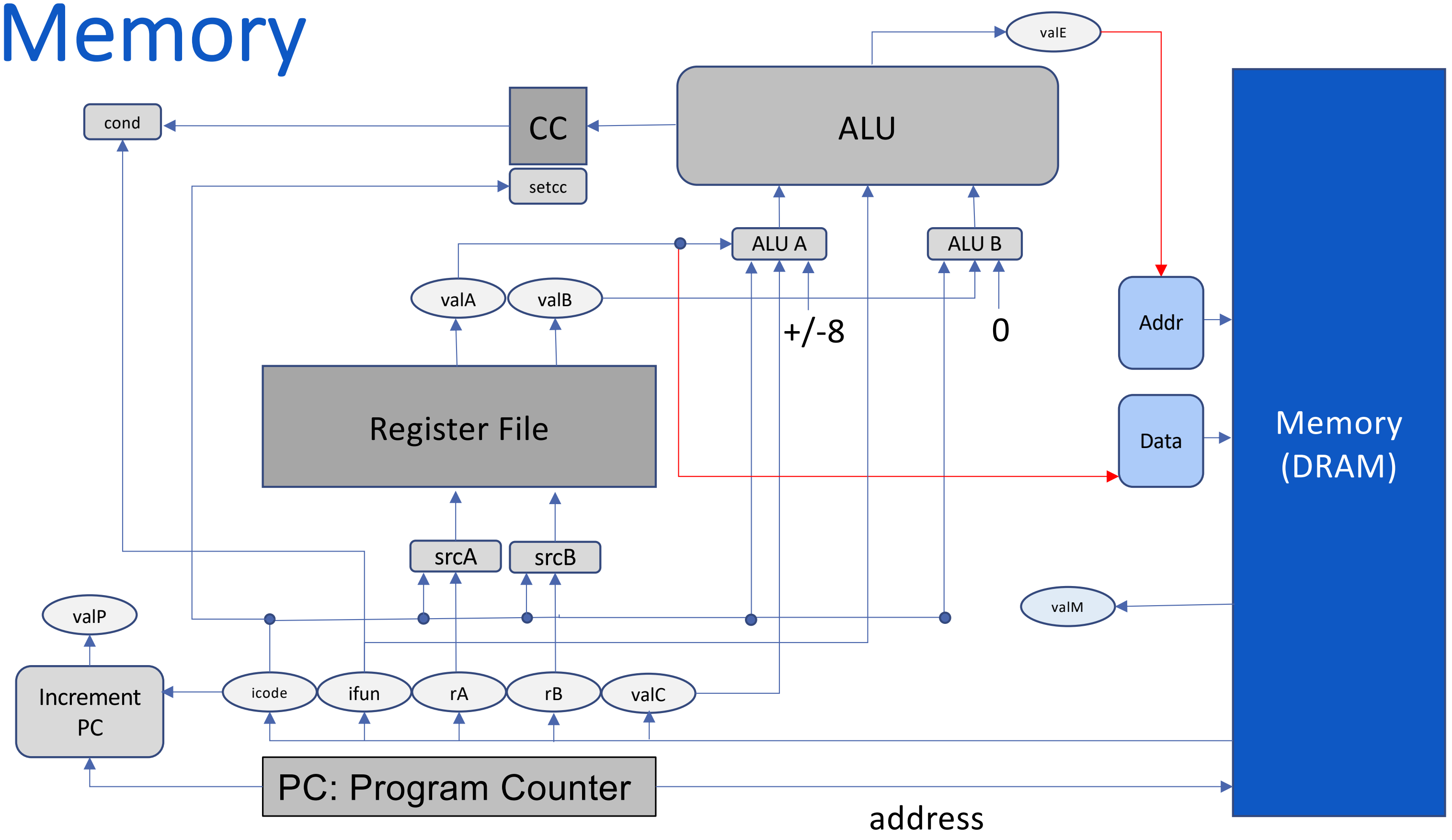
Implementing Execute



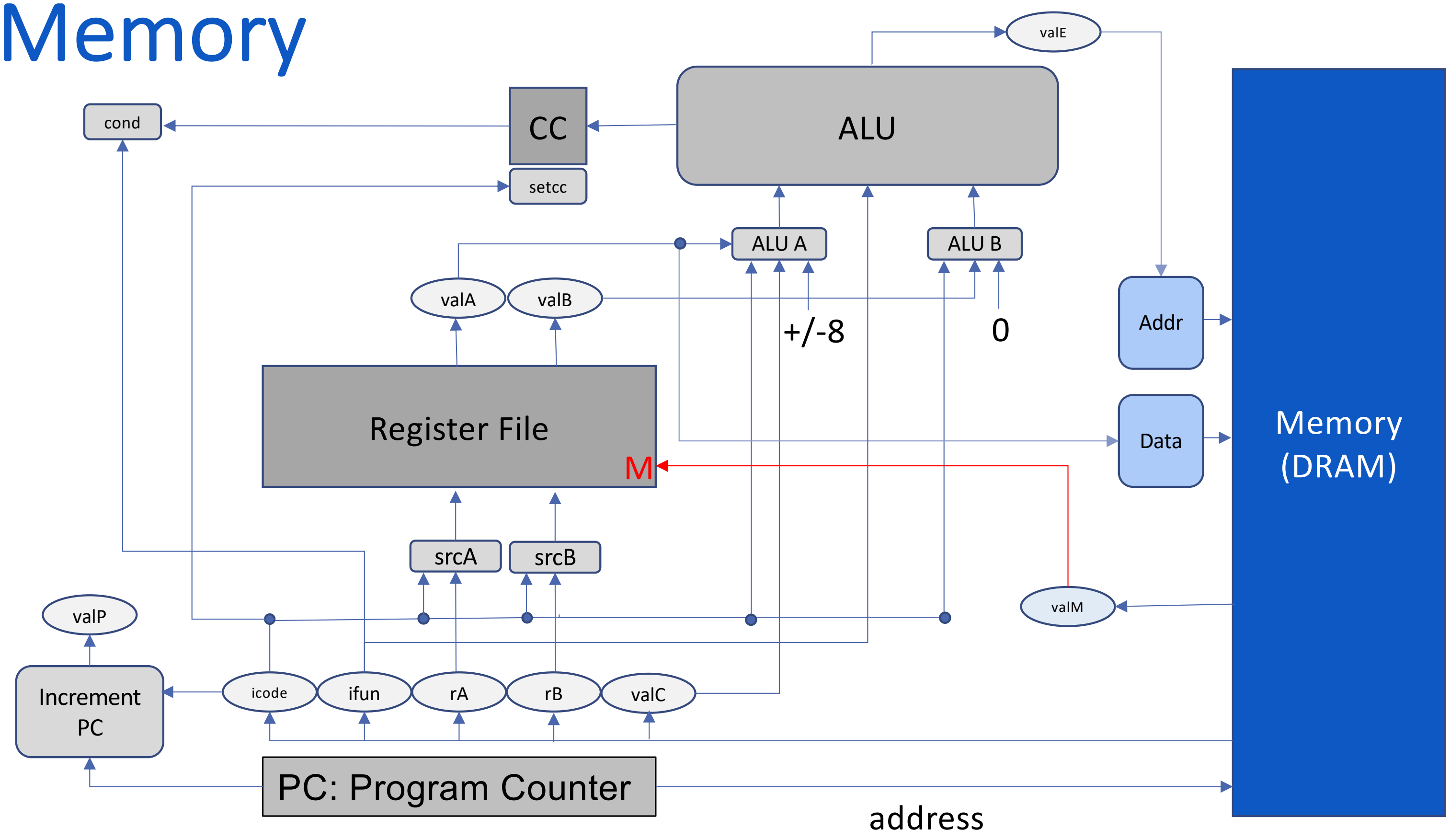
Memory

- What must we do in this stage:
  - Read from or write to memory
- What parts of the processor are involved:
  - Memory
  - valE (address)
  - valA (data)
  - valP (from the PC Increment)

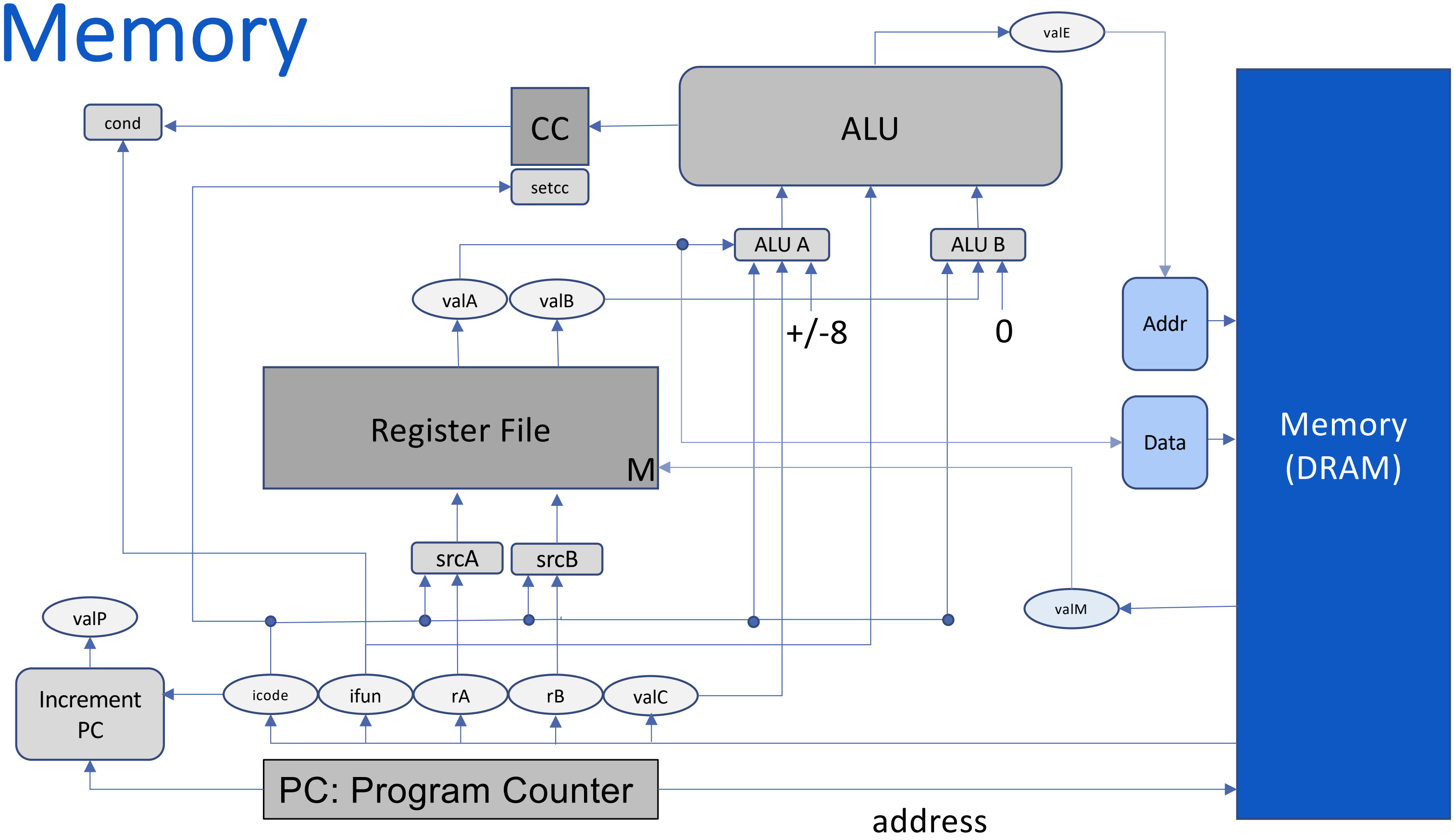
Memory


valE
cond
CC
ALU
setcc
ALU A
ALU B
valA
valB
+/-8
0
Addr
Register File
Data
Memory
(DRAM)
srcA
srcB
valP
valM
Increment
PC
icode
ifun
rA
rB
valC
PC: Program Counter
address

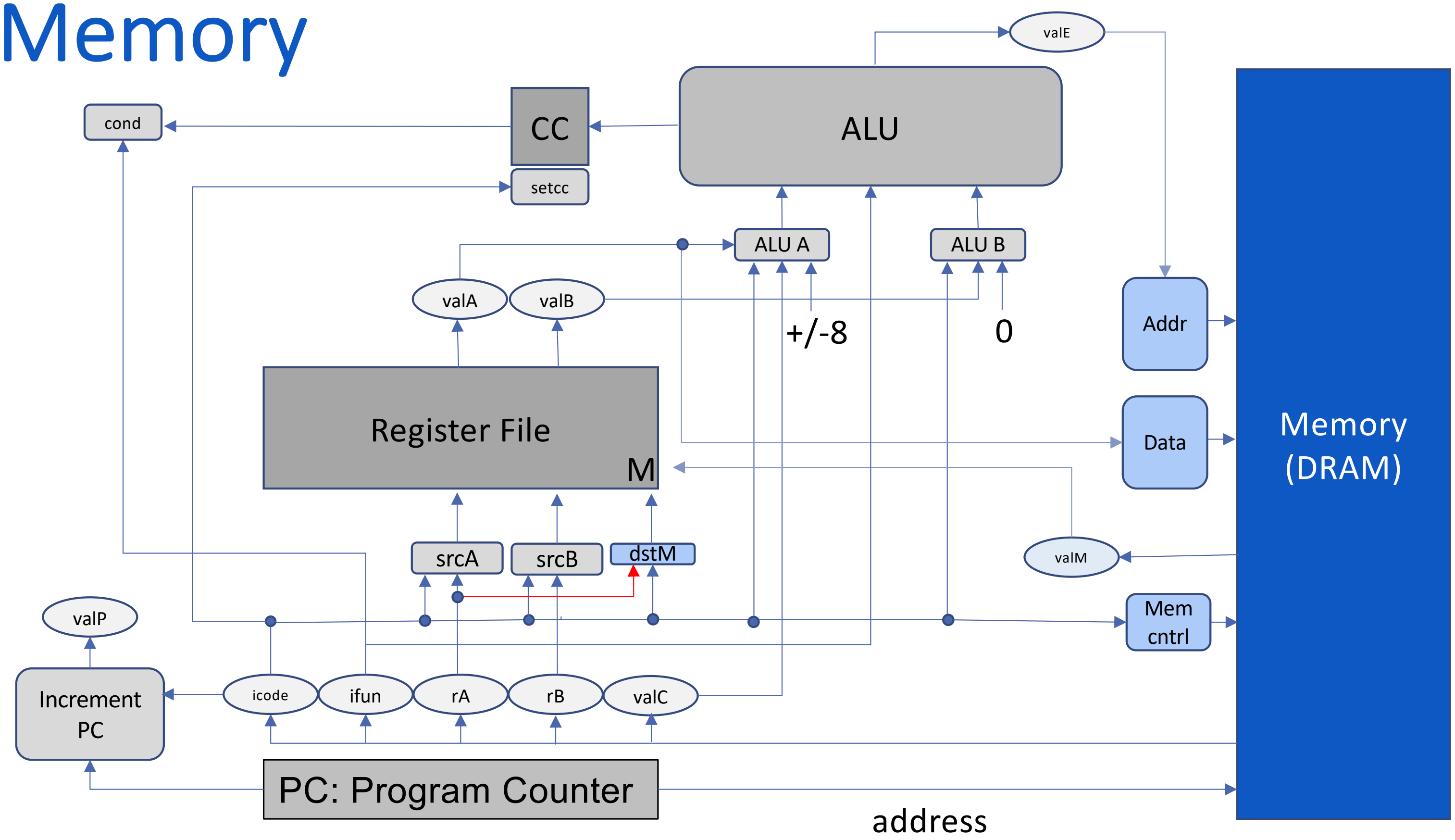
Memory


valE
cond
CC
ALU
setcc
ALU A
ALU B
valA
valB
+/-8
0
Addr
Register File
M
Data
Memory
(DRAM)
srcA
srcB
valP
valM
Increment
PC
icode
ifun
rA
rB
valC
PC: Program Counter
address

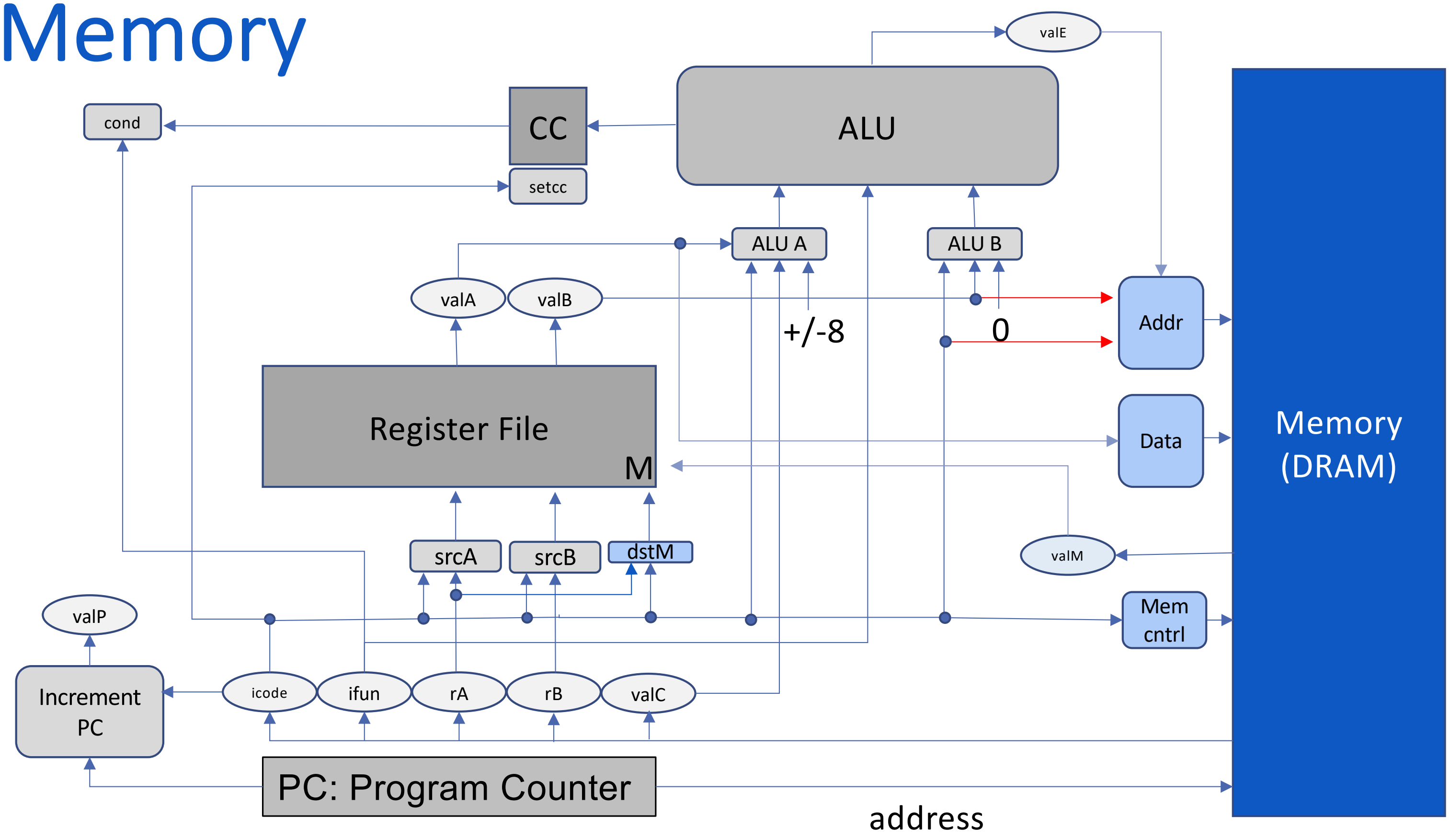
Memory


valE
cond
CC
ALU
setcc
ALU A
ALU B
valA
valB
+/-8
0
Addr
Register File
M
Data
Memory
(DRAM)
srcA
srcB
valP
valM
Increment
PC
icode
ifun
rA
rB
valC
PC: Program Counter
address

Memory

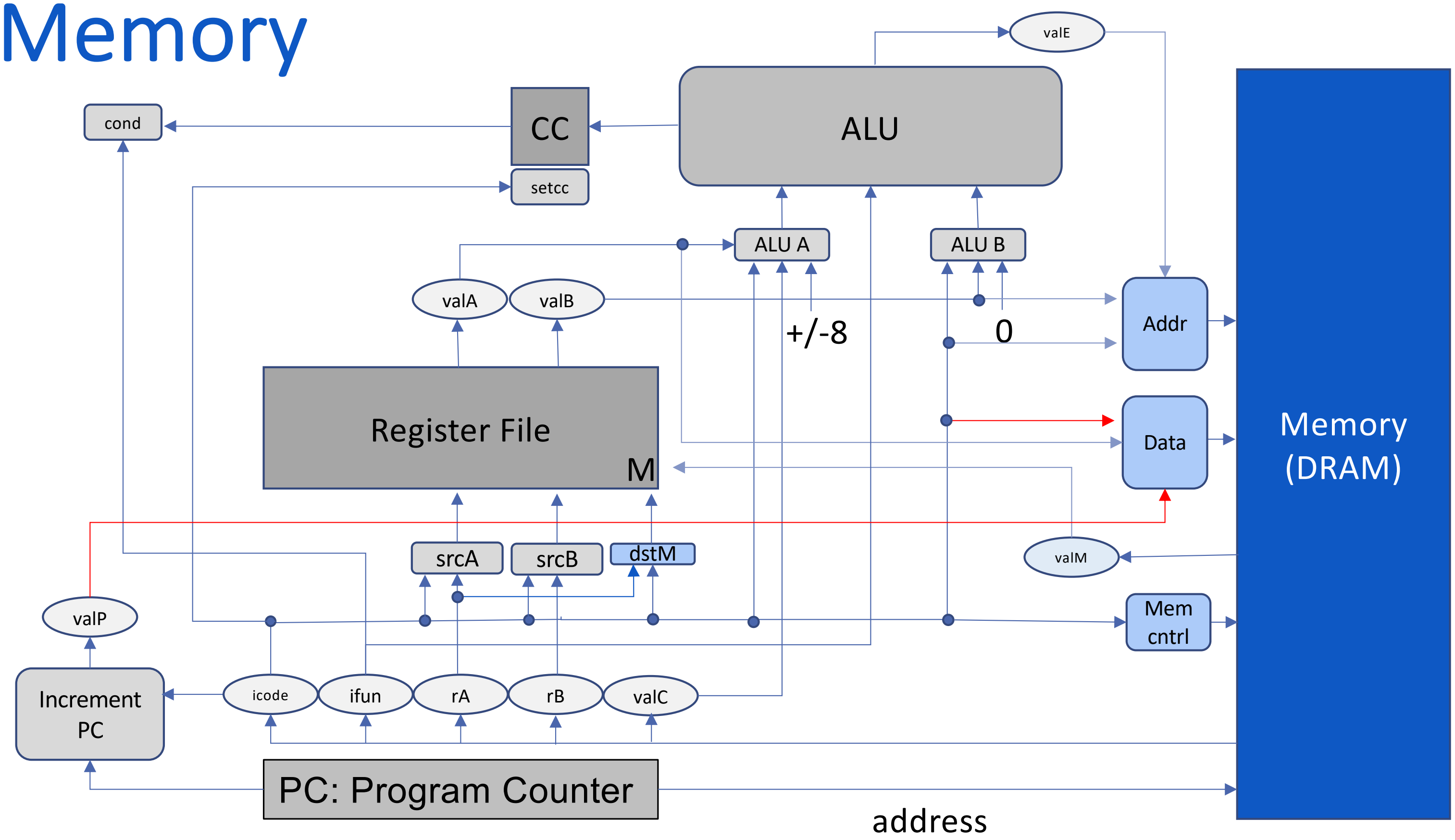

valE
cond
CC
ALU
setcc
ALU A
ALU B
valA
valB
+/-8
0
Addr
Register File
M
Data
Memory
(DRAM)
srcA
srcB
dstM
valM
valP
Mem
cntrl
Increment
PC
icode
ifun
rA
rB
valC
PC: Program Counter
address

Memory



valE
cond
CC
ALU
setcc
ALU A
ALU B
valA
valB
+/-8
0
Addr
Data
Memory
(DRAM)
Register File
M
valM
srcA
srcB
dstM
valP
Mem
cntrl
Increment
PC
icode
ifun
rA
rB
valC
PC: Program Counter
address

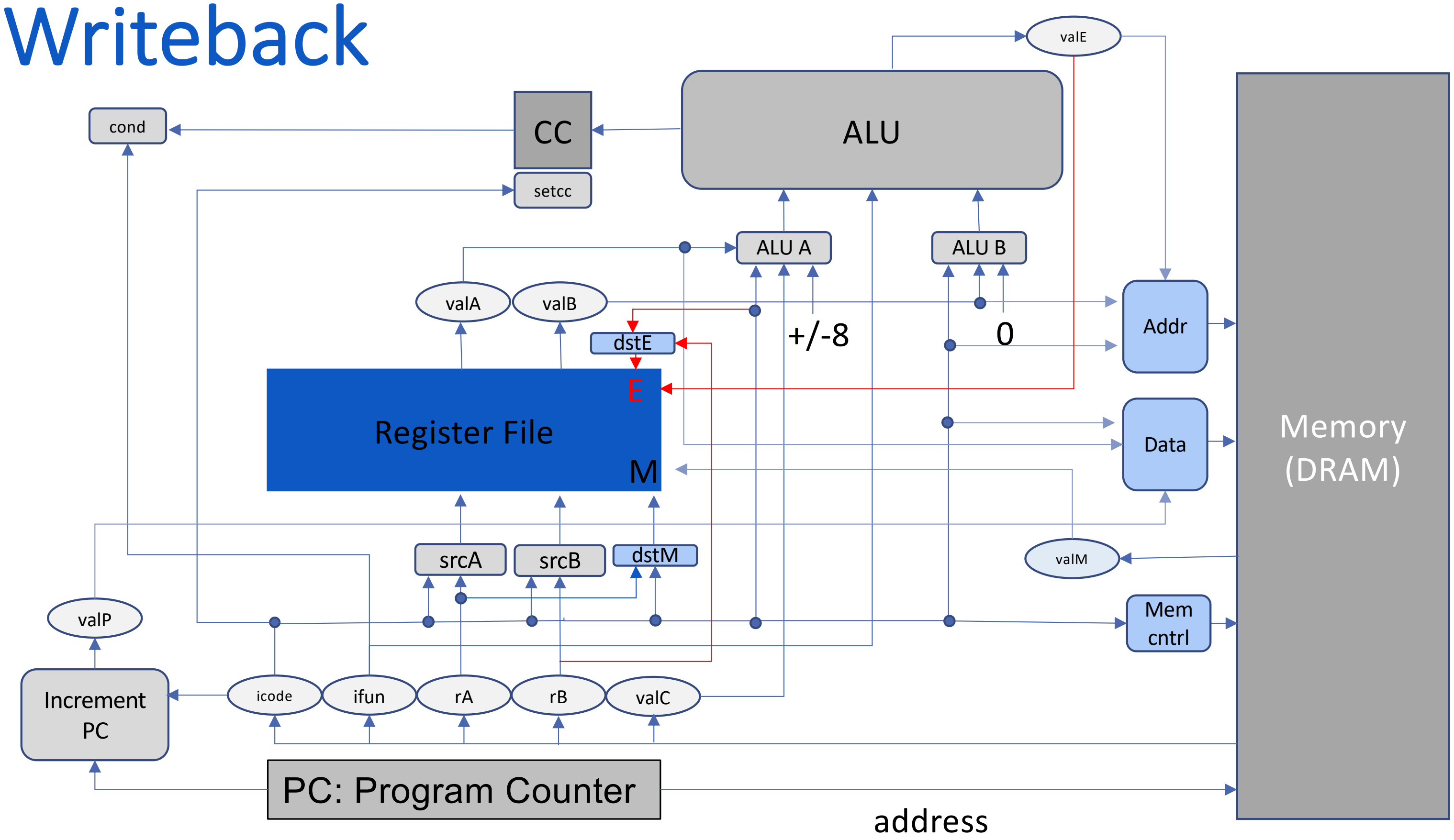
Memory


valE
cond
CC
ALU
setcc
ALU A
ALU B
valA
valB
+/-8
0
Addr
Register File
M
Data
Memory
(DRAM)
srcA
srcB
dstM
valM
valP
Mem
cntrl
Increment
PC
icode
ifun
rA
rB
valC
PC: Program Counter
address

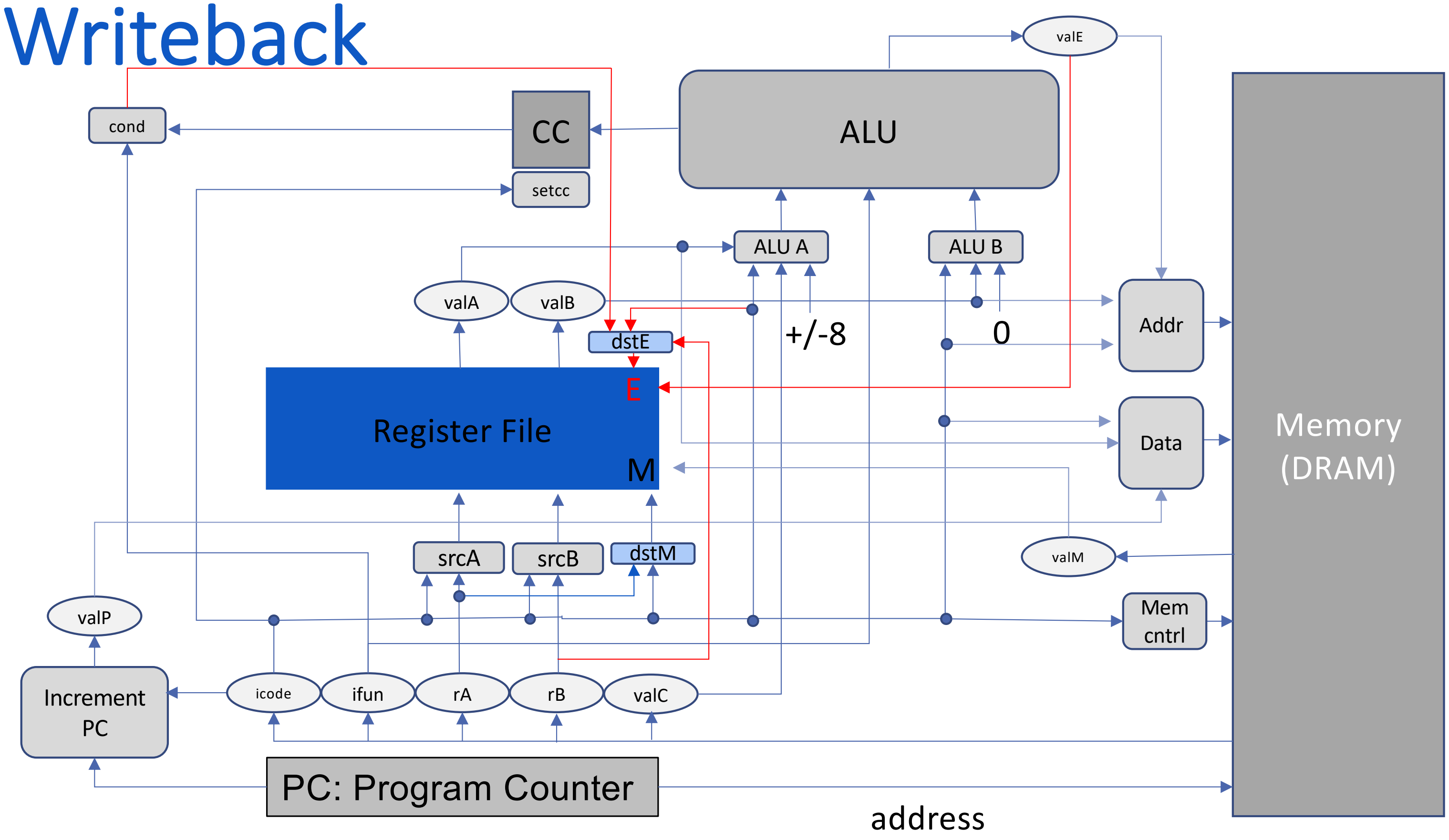
Writeback

- What must we do in this stage:
  - Write values into the register file
- What parts of the processor are involved:
  - Register file
  - valM (from memory)
  - valE (from the ALU)
  - cond (conditional moves!)

Writeback


valE
cond
CC
ALU
setcc
ALU A
ALU B
valA
valB
dstE
+/-8
0
Addr
E
Register File
M
Data
Memory
(DRAM)
srcA
srcB
dstM
valM
valP
Mem
cntrl
Increment
PC
icode
ifun
rA
rB
valC
PC: Program Counter
address

Writeback

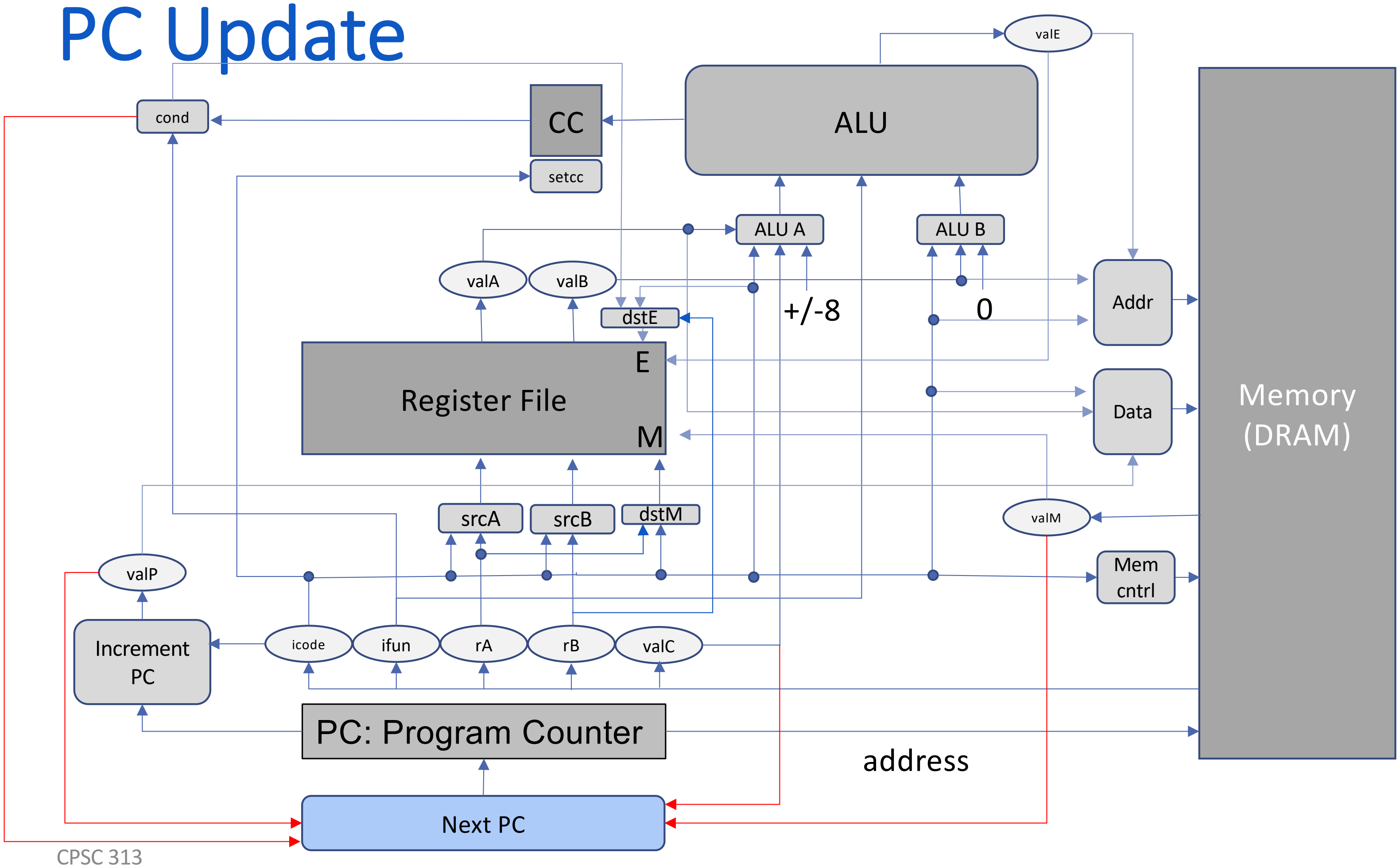


valE
cond
CC
ALU
setcc
ALU A
ALU B
valA
valB
dstE
+/-8
0
Addr
E
Register File
M
Data
Memory
(DRAM)
valM
srcA
srcB
dstM
valP
Mem
cntrl
Increment
PC
icode
ifun
rA
rB
valC
PC: Program Counter
address

PC Update

- What must we do in this stage:
  - Identify the address of the next instruction to execute
- What parts of the processor are involved:
  - PC
  - valP (normal case)
  - valM (ret)
  - valC (call, jmp)
  - cond

PC Update


valE
cond
CC
ALU
setcc
ALU A
ALU B
valA
valB
dstE
+/-8
0
Addr
E
Register File
M
Data
Memory
(DRAM)
srcA
srcB
dstM
valM
valP
Mem
cntrl
Increment
PC
icode
ifun
rA
rB
valC
PC: Program Counter
address
Next PC