Fetch Stage

Byte	0	1	2	3	4	5	6	7	8	9
rmmovq rA, D(rB)	4	0	rA	rB	D					
mrmovq D(rB), rA	5	0	rA	rB	D					

Instructions	Fetch
rrmovq rA, rB	icode:ifun ← M1[PC] rA:rB ← M1[PC+1] valP ← PC+10
irmovq V, rB	icode:ifun ← M1[PC] valC ← M8[PC+2] rA:rB ← M1[PC+1] valP ← PC+10
rmmovq rA, D(rB)	icode:ifun ← M1[PC] rA:rB ← M1[PC+1] valC ← M8[PC + 2] valP ← PC + 10
mrmovq D(rB), rA	icode:ifun ← M1[PC] rA:rB ← M1[PC+1] valC ← M8[PC + 2] valP ← PC + 10
OPq rA, rB	icode:ifun ← M1[PC] rA:rB ← M1[PC+1] valP ← PC+2
jXX Dest	icode:ifun ← M1[PC] valP ← PC+9 valC ← M8[PC+1]
cmovXX rA, rB	icode:ifun ← M1[PC] rA:rB ← M1[PC+1] valP ← PC+2
call Dest	icode:ifun ← M1[PC] valP ← PC+9 valC ← M8[PC+1]
ret	icode:ifun ← M1[PC] valP ← PC+1
pushq rA	icode:ifun ← M1[PC] valP ← PC+2 rA:rB ← M1[PC+1]
pop rA	icode:ifun ← M1[PC] valP ← PC+2 rA:rB ← M1[PC+1]

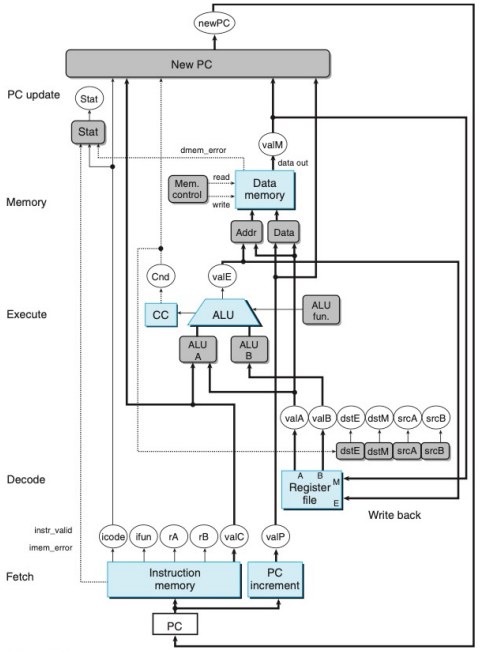

Figure 4.23 **Hardware structure of SEQ, a sequential implementation.** Some of the control signals, as well as the register and control word connections, are not shown.

Decode Stage

Byte	0	1	2	3	4	5	6	7	8	9
rmmovq rA, D(rB)	4	0	rA	rB	D					
mrmovq D(rB), rA	5	0	rA	rB	D					

Instructions	Decode
rrmovq rA, rB	valA ← R[rA] valB ← R[rB]
irmovq V, rB	X
rmmovq rA, D(rB)	valA ← R[rA] valB ← R[rB]
mrmovq D(rB), rA	valB ← R[rB]
OPq rA, rB	valA ← R[rA] valB ← R[rB]
jXX Dest	X
cmovXX rA, rB	valA ← R[rA] valB ← 0
call Dest	valB ← R[%rsp]
ret	valA ← R[%rsp] valB ← R[%rsp]
pushq rA	valA ← R[rA] valB ← R[%rsp]
pop rA	valA ← R[%rsp] valB ← R[%rsp]

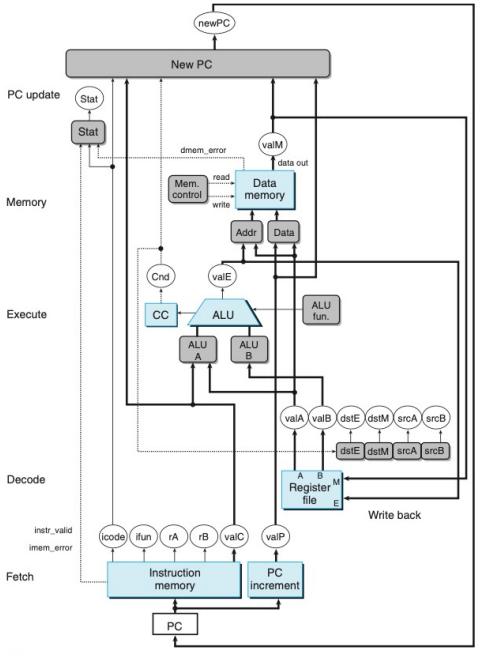

Figure 4.23 **Hardware structure of SEQ, a sequential implementation.** Some of the control signals, as well as the register and control word connections, are not shown.

Execute Stage

Byte	0	1	2	3	4	5	6	7	8	9
rmmovq rA, D(rB)	4	0	rA	rB	D					
mrmovq D(rB), rA	5	0	rA	rB	D					

Instructions	Execute
rrmovq rA, rB	valE ← valB + valC
irmovq V, rB	valE ← valB + valC
rmmovq rA, D(rB)	valE ← valB + valC
mrmovq D(rB), rA	valE ← valB + valC
OPq rA, rB	valE ← valB OP valA Set CC
jXX Dest	Cnd ← cond (CC, ifun)
cmovXX rA, rB	valE ← valB + valA If ! cond (CC, ifun) rB ← 0xF
call Dest	valE ← valB + -8
ret	valE ← valB + 8
pushq rA	valE ← val + -8
pop rA	valE ← valB + 8

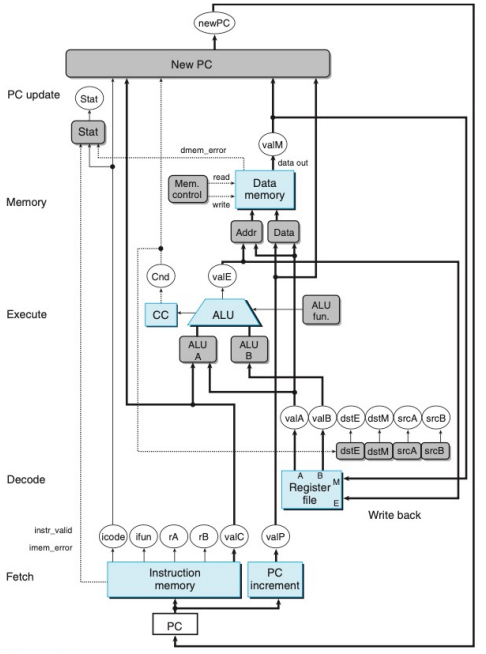

Figure 4.23 **Hardware structure of SEQ, a sequential implementation.** Some of the control signals, as well as the register and control word connections, are not shown.

Memory Stage

Byte	0	1	2	3	4	5	6	7	8	9
rmmovq rA, D(rB)	4	0	rA	rB	D					
mrmovq D(rB), rA	5	0	rA	rB	D					

Instructions	Memory
rrmovq rA, rB	
irmovq V, rB	
rmmovq rA, D(rB)	M_8[valE] ← valA
mrmovq D(rB), rA	valM ← M_8[valE]
OPq rA, rB	
jXX Dest	
cmovXX rA, rB	
call Dest	M_8[valE] ← valP
ret	valM ← M_8[valA]
pushq rA	M_8[valE] ← valA
pop rA	valM ← M_8[valA]

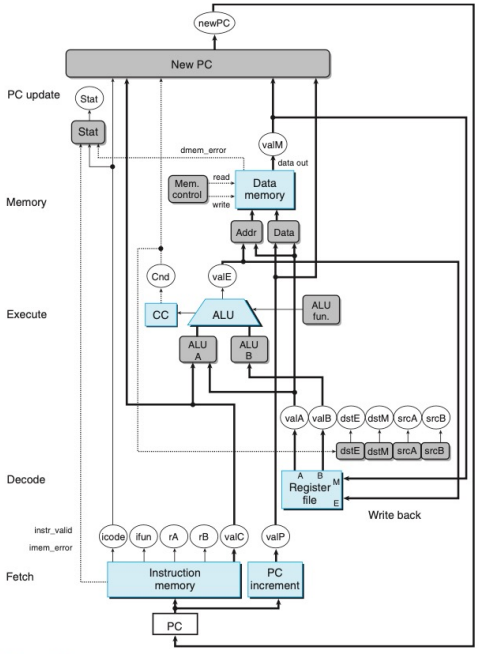

Figure 4.23 **Hardware structure of SEQ, a sequential implementation.** Some of the control signals, as well as the register and control word connections, are not shown.

Write back Stage

Byte	0	1	2	3	4	5	6	7	8	9
rmmovq rA, D(rB)	4	0	rA	rB	D					
mrmovq D(rB), rA	5	0	rA	rB	D					

Instructions	Write back
rrmovq rA, rB	R[rB] ← valE
irmovq V, rB	R[rB] ← valE
rmmovq rA, D(rB)	
mrmovq D(rB), rA	R[rA] ← valM
OPq rA, rB	R[rB] ← valE
jXX Dest	X
cmovXX rA, rB	R[rB] ← valE
call Dest	R[%rsp] ← valE
ret	R[%rsp] ← valE
pushq rA	R[%rsp] ← valE
pop rA	R[%rsp] ← valE R[rA] ← valM

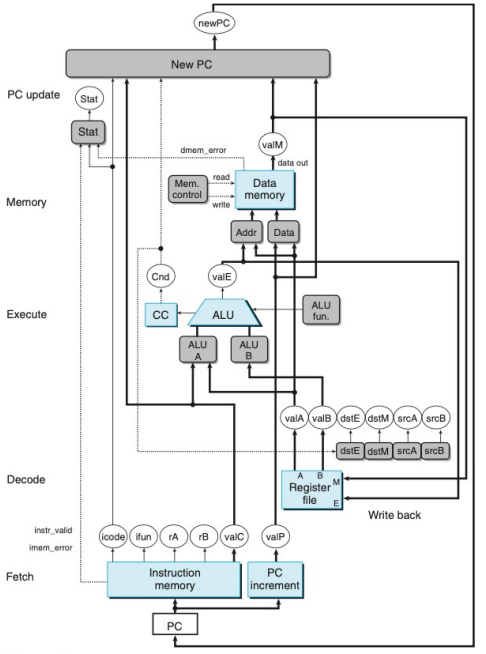

Figure 4.23 **Hardware structure of SEQ, a sequential implementation.** Some of the control signals, as well as the register and control word connections, are not shown.

PC update Stage

Byte	0	1	2	3	4	5	6	7	8	9
rmmovq rA, D(rB)	4	0	rA	rB	D					
mrmovq D(rB), rA	5	0	rA	rB	D					

Instructions	PC update
rrmovq rA, rB	PC ← valP
irmovq V, rB	PC ← valP
rmmovq rA, D(rB)	PC ← valP
mrmovq D(rB), rA	PC ← valP
OPq rA, rB	PC ← valP
jXX Dest	PC ← Cnd ? valC : valP
cmovXX rA, rB	PC ← valP
call Dest	PC ← valC
ret	PC ← valM
pushq rA	PC ← valP
pop rA	PC ← valP

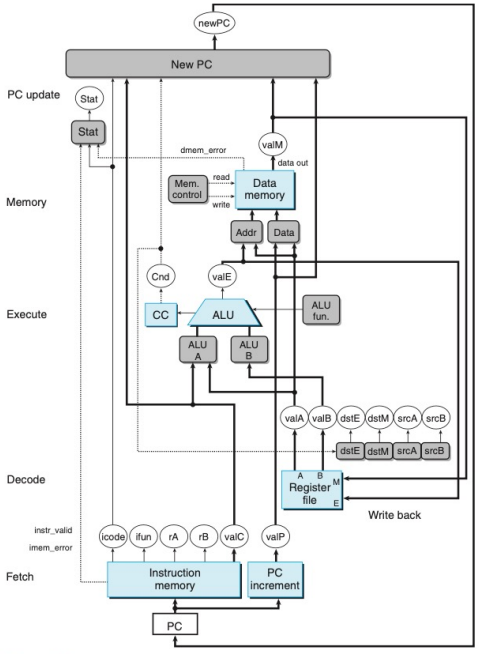

Figure 4.23 **Hardware structure of SEQ, a sequential implementation.** Some of the control signals, as well as the register and control word connections, are not shown.