486 Architecture

and SEQ organization

Byte	0	1	2	3	4	5	6	7	8	9
halt	0 0									
nop	1 0									
rrmovq rA, rB	2 0	rA rB								
irmovq V, rB	3 0	F rB	V							
rmmovq rA, D(rB)	4 0	rA rB	D							
mrmovq D(rB), rA	5 0	rA rB	D							
OPq rA, rB	6 fn	rA rB								
jXX Dest	7 fn	Dest								
cmovXX rA, rB	2 fn	rA rB								
call Dest	8 0	Dest								
ret	9 0									
pushq rA	A 0	rA F								
popq rA	B 0	rA F								

Operations

addq	6	0
subq	6	1
andq	6	2
xorq	6	3

Branches

jmp	7	0	jne	7	4
jle	7	1	jge	7	5
jl	7	2	jg	7	6
je	7	3			

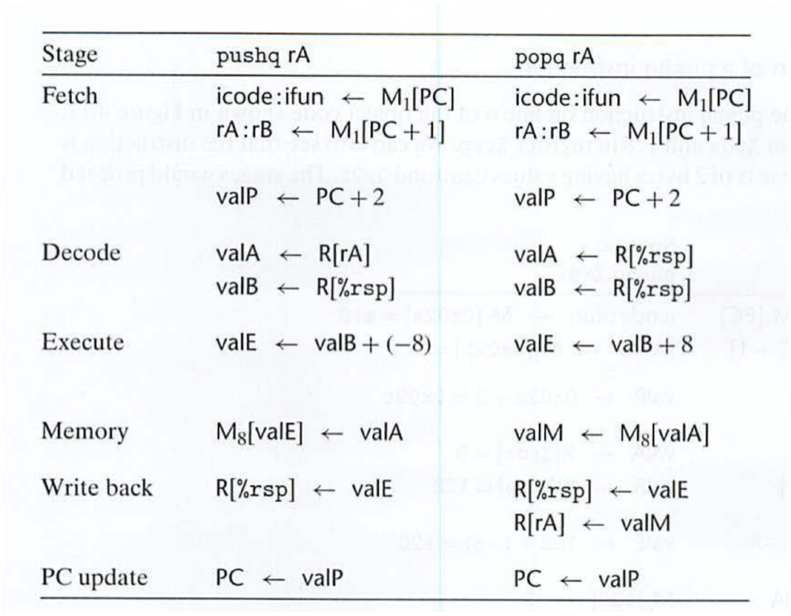
Moves

rrmovq	2	0	cmovne	2	4
cmovle	2	1	cmovge	2	5
cmovl	2	2	cmovg	2	6
cmove	2	3			

Number	Register name	Number	Register name
0	%rax	8	%r8
1	%rcx	9	%r9
2	%rdx	A	%r10
3	%rbx	B	%r11
4	%rsp	C	%r12
5	%rbp	D	%r13
6	%rsi	E	%r14
7	%rdi	F	No register

Stage	OPq rA, rB	rrmovq rA, rB	irmovq V, rB
Fetch	icode:ifun $\leftarrow$ $M_1[PC]$ rA:rB $\leftarrow$ $M_1[PC+1]$ valP $\leftarrow$ $PC+2$	icode:ifun $\leftarrow$ $M_1[PC]$ rA:rB $\leftarrow$ $M_1[PC+1]$ valP $\leftarrow$ $PC+2$	icode:ifun $\leftarrow$ $M_1[PC]$ rA:rB $\leftarrow$ $M_1[PC+1]$ valC $\leftarrow$ $M_8[PC+2]$ valP $\leftarrow$ $PC+10$
Decode	valA $\leftarrow$ R[rA] valB $\leftarrow$ R[rB]	valA $\leftarrow$ R[rA]	
Execute	valE $\leftarrow$ valB OP valA Set CC	valE $\leftarrow$ 0 + valA	valE $\leftarrow$ 0 + valC
Memory			
Write back	R[rB] $\leftarrow$ valE	R[rB] $\leftarrow$ valE	R[rB] $\leftarrow$ valE
PC update	PC $\leftarrow$ valP	PC $\leftarrow$ valP	PC $\leftarrow$ valP

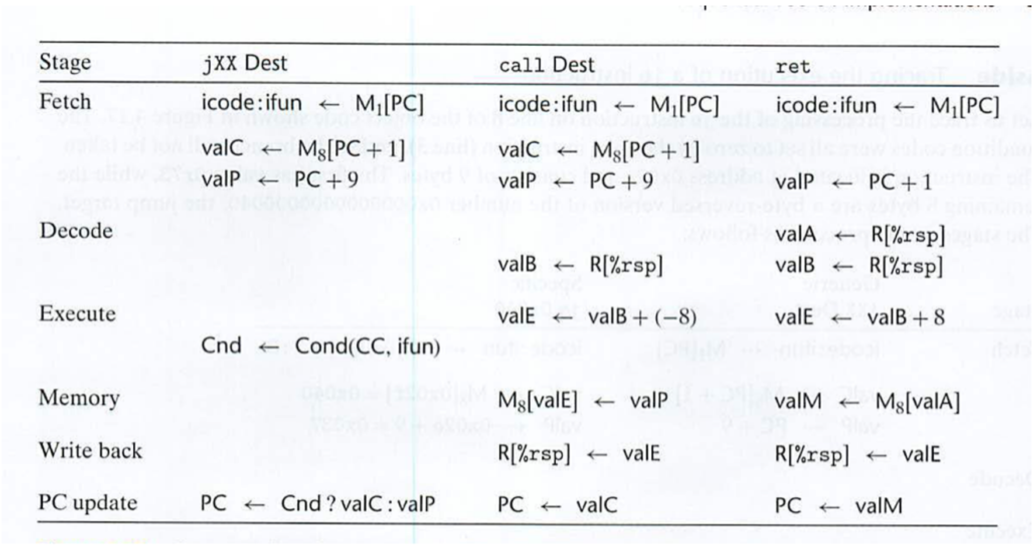
Stage	rmmovq rA, D(rB)	mrmovq D(rB), rA
Fetch	icode:ifun $\leftarrow$ $M_1[PC]$ rA:rB $\leftarrow$ $M_1[PC+1]$ valC $\leftarrow$ $M_8[PC+2]$ valP $\leftarrow$ $PC+10$	icode:ifun $\leftarrow$ $M_1[PC]$ rA:rB $\leftarrow$ $M_1[PC+1]$ valC $\leftarrow$ $M_8[PC+2]$ valP $\leftarrow$ $PC+10$
Decode	valA $\leftarrow$ R[rA] valB $\leftarrow$ R[rB]	 valB $\leftarrow$ R[rB]
Execute	valE $\leftarrow$ valB + valC	valE $\leftarrow$ valB + valC
Memory	M_8[valE] $\leftarrow$ valA	valM $\leftarrow$ M_8[valE]
Write back		R[rA] $\leftarrow$ valM
PC update	PC $\leftarrow$ valP	PC $\leftarrow$ valP



Stage	pushq rA	popq rA
Fetch	icode:ifun ← M_1[PC] rA:rB ← M_1[PC + 1] valP ← PC + 2	icode:ifun ← M_1[PC] rA:rB ← M_1[PC + 1] valP ← PC + 2

Decode	valA ← R[rA]
	valB ← R[%rsp]
		valA ← R[%rsp]
		valB ← R[%rsp]

Execute	valE ← valB + (−8)	valE ← valB + 8
Memory	M_8[valE] ← valA	valM ← M_8[valA]
Write back	R[%rsp] ← valE	R[%rsp] ← valE R[rA] ← valM
PC update	PC ← valP	PC ← valP

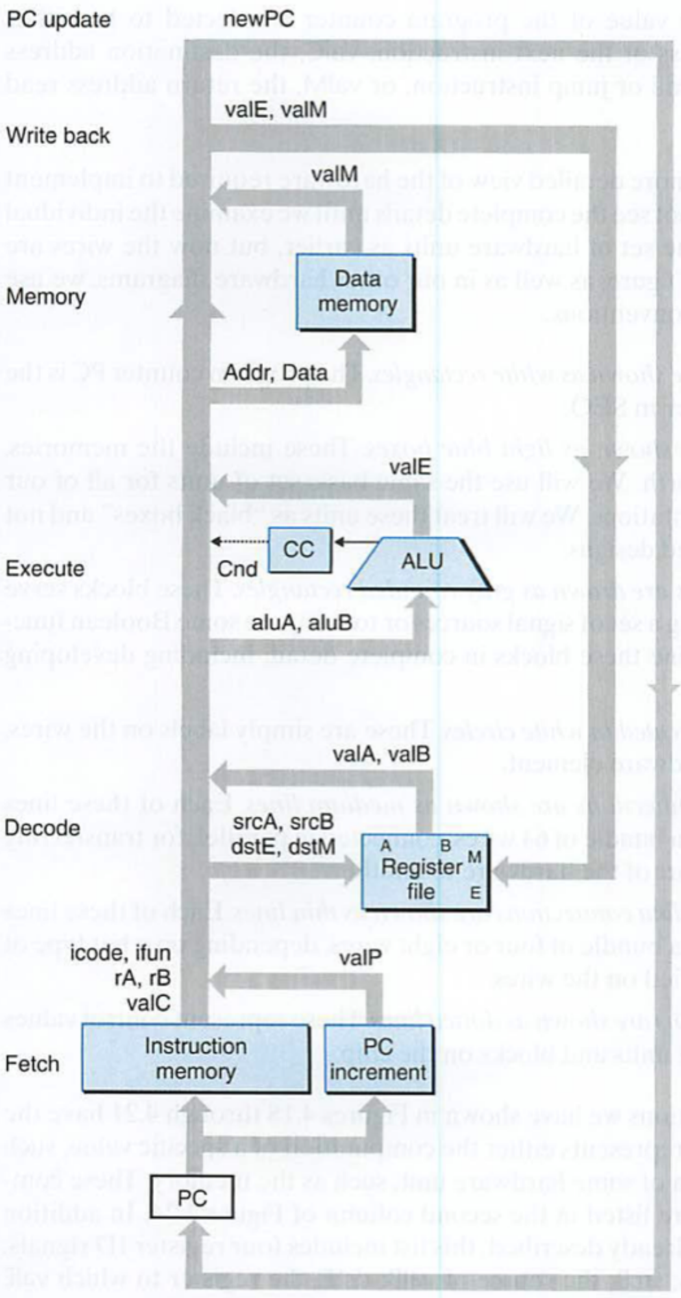


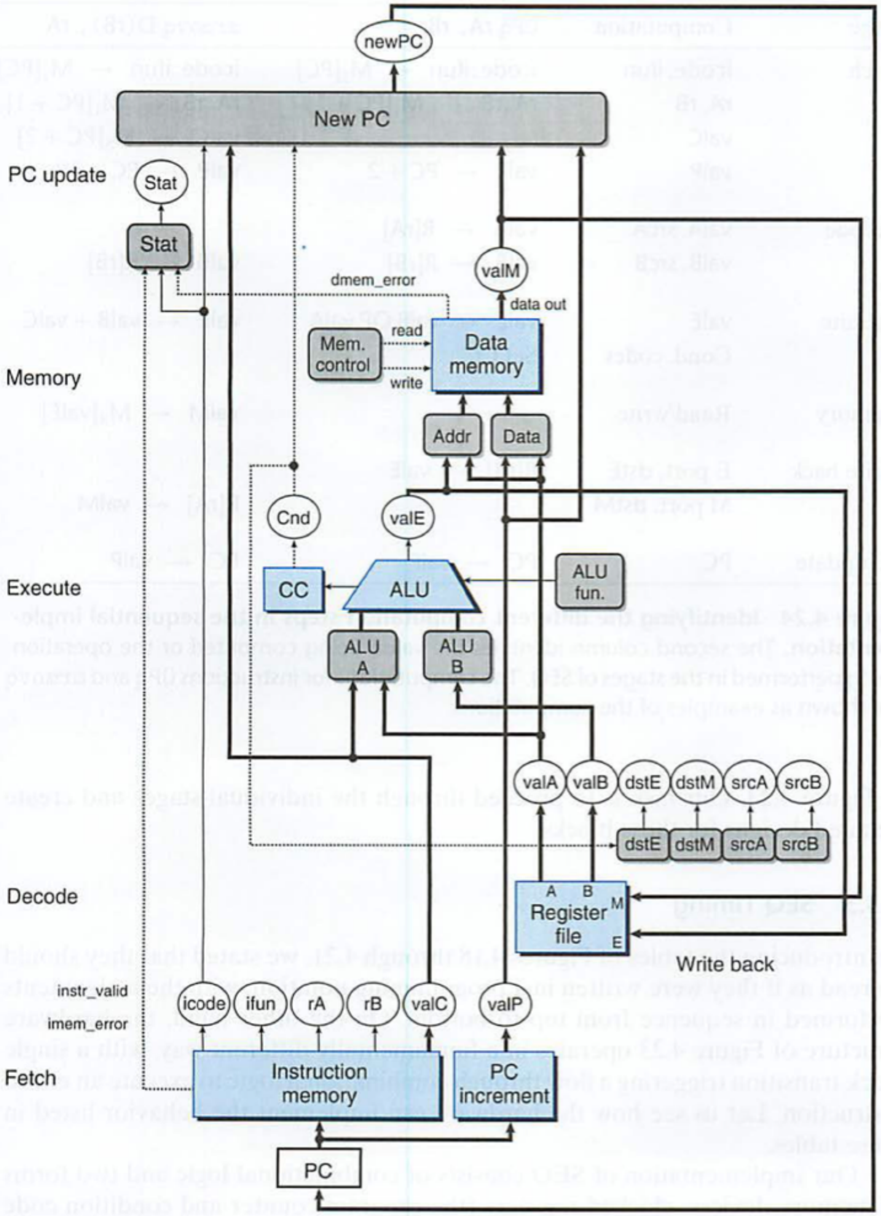
Stage	jXX Dest	call Dest	ret

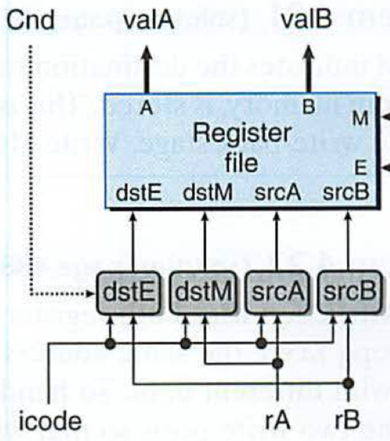
Fetch	icode:ifun ← M_1[PC]
	valC ← M_8[PC + 1]
	valP ← PC + 9
		icode:ifun ← M_1[PC]
		valC ← M_8[PC + 1]
		valP ← PC + 9
			icode:ifun ← M_1[PC]
			
			valP ← PC + 1

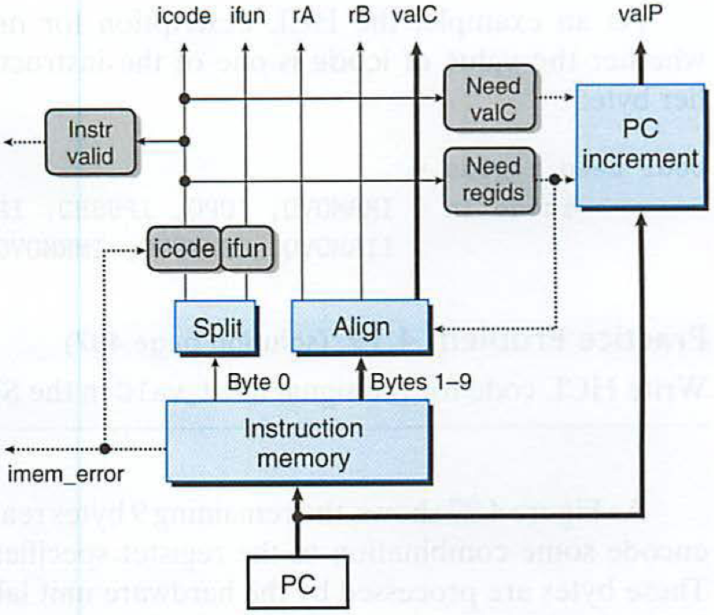
Decode		
		valB ← R[%rsp]
			valA ← R[%rsp]
			valB ← R[%rsp]

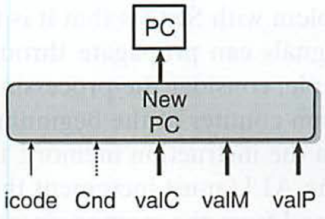
Execute	 Cnd ← Cond(CC, ifun)	valE ← valB + (−8)	valE ← valB + 8
Memory		M_8[valE] ← valP	valM ← M_8[valA]
Write back		R[%rsp] ← valE	R[%rsp] ← valE
PC update	PC ← Cnd ? valC : valP	PC ← valC	PC ← valM

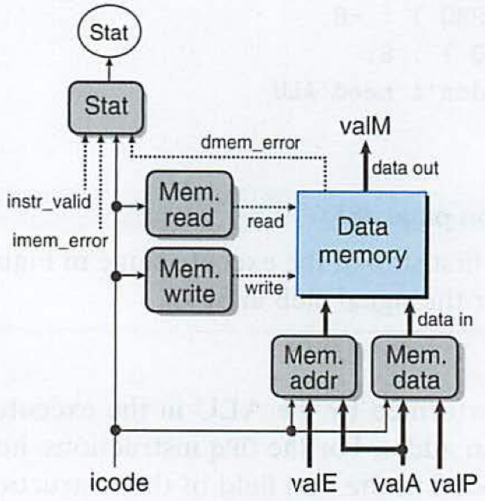

PC update
newPC
valE, valM
Write back
valM
Data memory
Memory
Addr, Data
valE
CC
ALU
Execute
Cnd
aluA, aluB
valA, valB
Decode
srcA, srcB
dstE, dstM
A
B
M
Register file
E
icode, ifun
rA, rB
valC
valP
Fetch
Instruction memory
PC increment
PC

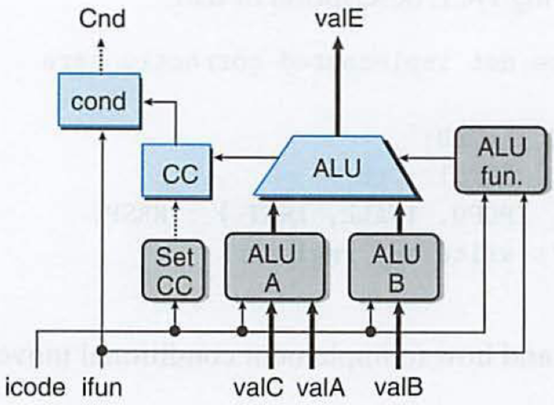

newPC
New PC
PC update
Stat
Stat
dmem_error
valM
data out
read
Mem. control
write
Data memory
Memory
Addr
Data
Cnd
valE
Execute
CC
ALU
ALU fun.
ALU A
ALU B
valA
valB
dstE
dstM
srcA
srcB
dstE
dstM
srcA
srcB
Decode
A
B
M
Register file
E
Write back
instr_valid
imem_error
icode
ifun
rA
rB
valC
valP
Fetch
Instruction memory
PC increment
PC


Cnd
valA
valB
valM
valE
A
B
M
Register file
E
dstE
dstM
srcA
srcB
dstE
dstM
srcA
srcB
icode
rA
rB


icode
ifun
rA
rB
valC
valP
Need valC
Instr valid
PC increment
Need regids
icode
ifun
Split
Align
Byte 0
Bytes 1–9
Instruction memory
imem_error
PC


PC
New PC
icode
Cnd
valC
valM
valP


Stat
Stat
valM
dmem_error
data out
instr_valid
Mem. read
read
Data memory
imem_error
Mem. write
write
data in
Mem. addr
Mem. data
icode
valE
valA
valP


Cnd
valE
cond
CC
ALU
ALU fun.
Set CC
ALU A
ALU B
icode
ifun
valC
valA
valB