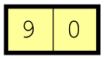
ret



Fetch icode : ifun ← M_1[PC]

Decode valA ← R[%rsp]

Execute valE ← valA + 8

Memory valM ← M_8[valP]

Write back R[%rsp] ← valE

PC update PC ← valM

cmovXX rA, rB

fetch icode : ifun ← M_1[PC]

rA : rB ← M_1[PC+1]

valP ← PC+2

decode valA ← R[rA]

execute Cnd ← Cond(CC, fn)

memory

Writeback if (Cnd) R[rB] ← valA

PC update PC ← valP

irmovq V, rB

i → R

Fetch icode : ifun ← M_1[PC]

rA : rB ← M_1[PC+1]

valC ← M_8[PC+2]

valP ← PC+10

Decode

Execute valE ← 0 + valC

Memory

Write back R[rB] ← valE

PC update PC ← valP

rmmovq rA, D(rB) (R→M)

Fetch icode:ifun ← M_1[PC]

rA:rB ← M_1[PC+1]

valC ← M_8[PC+2]

valP ← PC + 10

Decode valA ← R[rA]

valB ← R[rB]

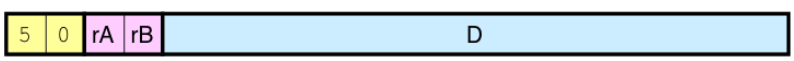
Execute valE ← valC + valB

Memory M_8[valE] ← valA → memory değerlerini değiştiriyoruz yes

Write back → register değerlerini değiştiriyoruz yes

PC update PC ← valP

mrmovq D(rB), rA (M→R)



fetch icode:ifun ← M_1[PC]

rA:rB ← M_1[PC+1]

valC ← M_1[PC+2]

valP ← PC + 10

decode valB ← R[rB]

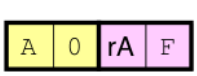
execute valE ← valC + valB

memory valM ← M_8[valE]

write back R[rA] ← valM

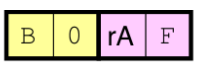
PC update PC ← valP

pushq rA



Fetch	icode : ifun ← M_1[PC]
	rA : rB ← M_1[PC+1]
	valP ← PC + 2
Decode	valA ← R[rA]
	valB ← R[%rsp]
Execute	valE ← valB - 8
Memory	M_8[valE] ← valA
Write back	R[%rsp] ← valE
PC update	PC ← valP

popq rA



fetch	icode : ifun ← M_1[PC]
	rA : rB ← M_1[PC+1]
	valP ← PC + 2
decode	valB ← R[%rsp]
execute	valE ← valB + 8
memory	valM ← M_8[valB]
write back	R[rA] ← valM
	R[%rsp] ← valE
PC update	PC ← valP