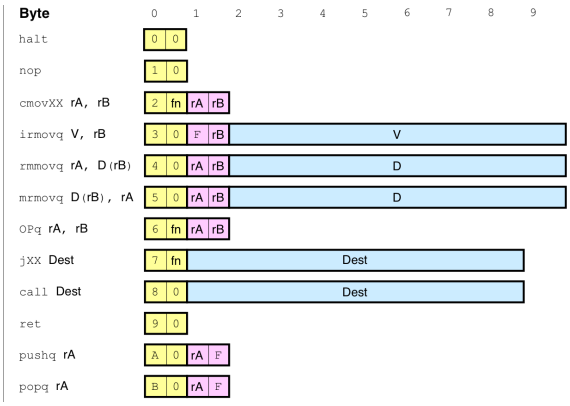


halt

Fetch:

icode : ifun ← M_1[PC]

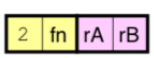
valP ← 1

Decode :

int Stat :

icode == IHALT : SHALT;

* cmovXX rA, rB



Fetch:

icode : ifun ← M_1[PC]

rA : rB ← M_1[PC+1]

valP ← PC+4

Decode:

valA ← R[rA]

valB ← 0

Execute :

valE ← valA + valB

if Cond (CC, ifun) , rB ← 0xF

Memory :

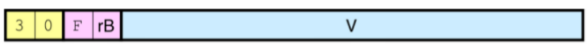
Write back :

if Cond(CC, ifun), R[rB] ← valE

PC Update :

PC ← valP

irmovq V, rB



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 :

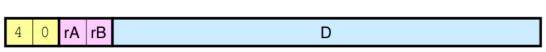
Write Back :

R[rB] ← valC

PC Update :

PC ← valP

rmmovq rA, D(rB)



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 ← valB + valC

Memory :

M[valE] ← valA

Write Back :

PC update :

PC ← valP

✳ `mrmovq D(rB), rA`

Fetch:

icode : ifun ← M_1[PC]

rA : rB ← M_1[PC+1]

valC ← M_8[PC+2]

valP ← PC+10

Decode:

valB ← R[rB]

Execute:

valE ← valC + valB

Memory:

valM ← M_8[valE]

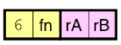
Write Back:

R[rA] ← valM

PC Update:

PC ← valP

✳ `OPq rA, rB`



Fetch:

icode : ifun ← M_1[PC]

rA : rB ← M_1[PC+1]

valP ← PC+2

Decode:

valA ← R[rA]

valB ← R[rB]

Execute:

valE ← valB OP valA

Set CC (cnd, ifun)

Memory:

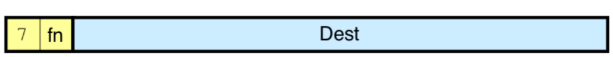
Write back:

R[rB] ← valE

PC Update

PC ← valP

✳ `jXX Dest`



Fetch:

icode : ifun ← M_1[PC]

valC ← M_8[PC+1]

valP ← PC+9

Decode:

Execute:

Cnd ← (CC, ifun)

Memory:

Write Back:

PC Update:

PC ← Cnd ? valC : valP

`call Dest`

Fetch :

icode ← - - -

valP ← PC+9

valC ← M_8[PC+1]

Decode :

valA ← R[%rsp]

valB ← R[%rsp]

Execute :

valE ← valA - 8

Memory

M[valE] ← valP

Write Back :

R[%rsp] ← valE

PC update :

PC ← valC

`ret`

F :

icode : ifun ← M_1[PC]

valP ← PC+1

D :

valA ← R[%rsp]

valB ← R[%rsp]

E :

valE ← valA+8

M :

valM ← M_8[valB]

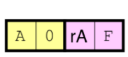
WB :

R[%rsp] ← valE

PC :

PC ← valM

`pushq rA`



F :

icode : ifun ← M_1[PC]

rA, rB ← M_1[PC+1]

valP ← PC+2

D :

valA ← R[rA]

valB ← R[rsp]

E : valE ← valB - 8

M : M_8[valE] ← valA

WB : R[rsp] ← valE

PC : PC ← valP

popq rA

F:

icode : ifun ← M_1[PC]

rA : rB ← M_1[PC+1]

valP ← PC+2

D:

valB ← R[rsp]

E:

valE ← valB+8

M:

valM ← M_8[valB]

WB:

R[rsp] ← valE

R[rA] ← valM

PC:

PC ← valP