# Y86-64微指令记忆卡片

| 阶段 | Opq rA, rB | rrmovq rA, rB | irmovq V, rB | rmmovq rA, D(rB) | mrmovq D(rB), rA |
|---|---|---|---|---|---|
| 取指 | icode:ifun <- M1[PC]<br>rA:rB <- M1[PC+1]<br>valP <- PC+2 | icode:ifun <- M1[PC]<br>rA:rB <- M1[PC+1]<br>valP <- PC+2 | icode : ifun <- M1[PC]<br>rA:rB <- M1[PC+1]<br>valC <- M8[PC+2]<br>valP <- PC+10 | icode:ifun <- M1[PC]<br>rA:rB <- M1[PC+1]<br>valC <- M8[PC+2]<br>valP <- PC+10 | icode:ifun <- M1[PC]<br>rA:rB <- M1[PC+1]<br>valC <- M8[PC+2]<br>valP <- PC+10 |
| 译码 | valA <- R[rA]<br>valB <- R[rB] | valA <- R[rA] | | valA <- R[rA]<br>valB <- R[rB] | valB <- R[rB] |
| 执行 | valE <- valB OP valA<br>Set CC(设置条件码) | valE <- 0+valA | valE <- 0 + valC | valE <- valB + valC | valE <- valB + valC |
| 访存 | | | | M8[valE] <- valA | valM <- M8[valE] |
| 写回 | R[rB] <- valE | R[rB] <- valE | R[rB] <- valE | | R[rA] <- valM |
| 更新PC | PC <- valP | PC <- valP | PC <- valP | PC <- valP | PC <- valP |

valA用来获取第一寄存器值

valB用来获取第二寄存器值

valE和设置操作码

| 阶段 | pushq rA | popq rA | jXX Dest | cmovXX rA, rB | call Dest | ret |
|---|---|---|---|---|---|---|
| 取指 | icode:ifun <- M1[PC]<br>rA:rB <- M1[PC+1]<br>valP <- PC+2 | icode:ifun <- M1[PC]<br>rA:rB <- M1[PC+1]<br>valP <- PC+2 | icode:ifun <- M1[PC]<br>valC <- M8[PC+1]<br>valP <- PC+9 | icode:ifun <- M1[PC]<br>rA:rB <- M1[PC+1]<br>valP <- PC+2 | icode:ifun <- M1[PC]<br>valC <- M8[PC+1]<br>valP <- PC+9 | icode:ifun <- M1[PC]<br>valP <- PC+1 |
| 译码 | valA <- R[rA]<br>valB <- R[%rsp] | valA <- R[%rsp]<br>valB <- R[%rsp] | | valA <- R[rA] | valB <- R[%rsp] | valA <- R[%rsp]<br>valB <- R[%rsp] |
| 执行 | valE <- valB+(-8) | valE <- valB+8 | Cnd <- Cond(CC, ifun) | valE <- 0+valA<br>Cnd <- Cond(CC, ifun) | valE <- valB+(-8) | valE <- valB+8 |
| 访存 | M8[valE] <- valA | valM <- M8[valA] | | | M8[valE] <- valP | valM <- M8[valA] |
| 写回 | R[%rsp] <- valE | R[%rsp] <- valE<br>R[rA] <- valM | | if(Cnd)   R[rB] <- valE | R[%rsp] <- valE | R[%rsp] <- valE |
| 更新PC | PC <- valP | PC <-valP | PC <- Cnd?valC:valP | PC <-valP | PC <- valC | PC <- valM |

valA用来读内存

valB用来操作栈指针

valE和设置操作码

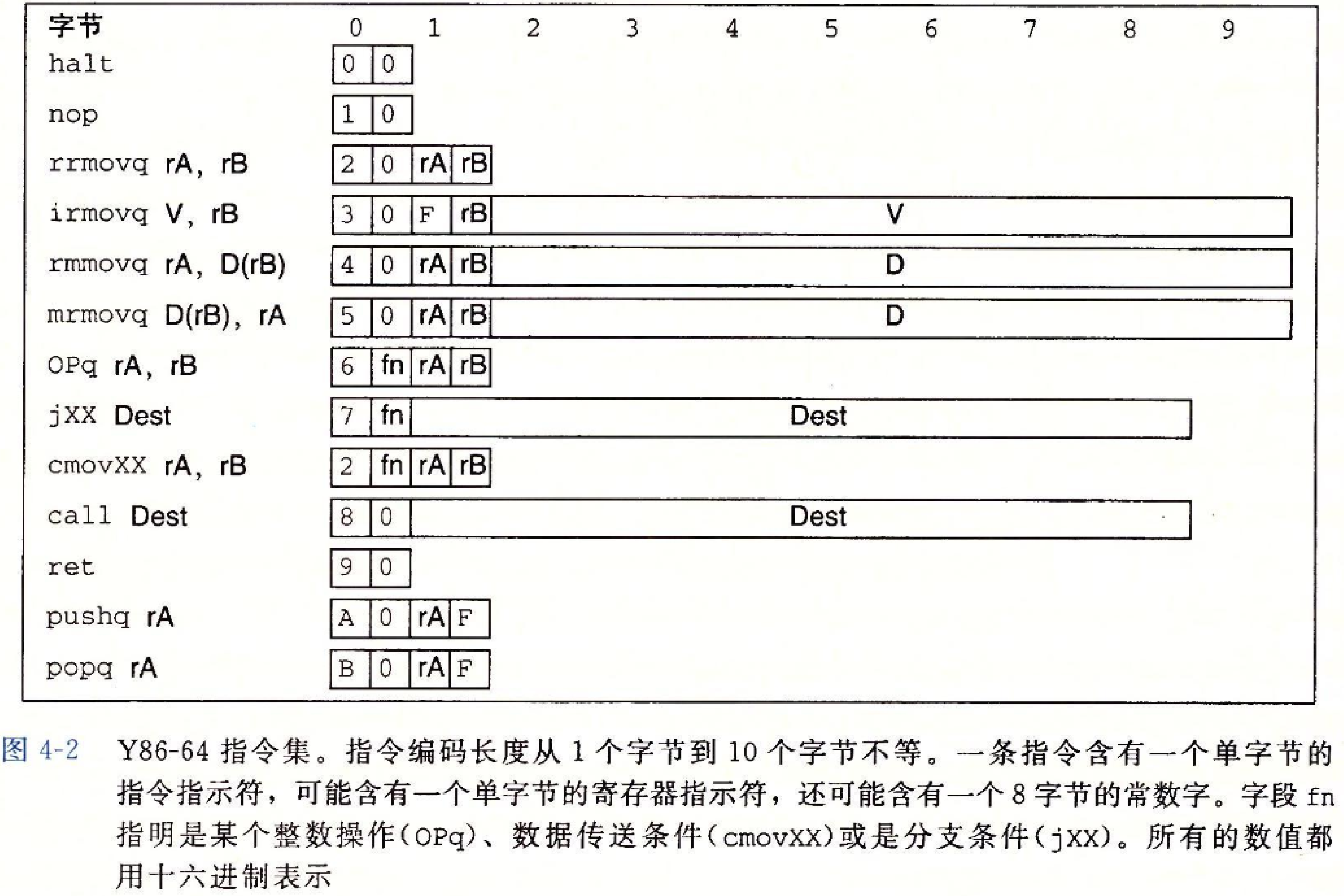
入栈

出栈

入栈先减小栈指针，再存数

出栈先取数，再增加栈指针

# Y86-64微指令机器码格式



| 字节 | 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 | | | | | | | | |

图 4-2　Y86-64 指令集。指令编码长度从 1 个字节到 10 个字节不等。一条指令含有一个单字节的指令指示符，可能含有一个单字节的寄存器指示符，还可能含有一个 8 字节的常数字。字段 fn 指明是某个整数操作(OPq)、数据传送条件(cmovXX)或是分支条件(jXX)。所有的数值都用十六进制表示