x86 指令

- `addl $0xDEADBEEF, -0x3f(%eax,%esi,4)`

1. 指令格式为 `addl imm32, mem`，对应 Opcode 为 `81 /0`；
2. 内存寻址：Base(%eax), Index(%esi), Scale(4), Displacement(-0x3F=0xC1)；
3. ModR/M：`mod=01`（8位位移），`reg=000`（源操作数`%eax`），`r/m=100`（需要SIB），ModR/M=01000100=0x44；
4. SIB：`scale=10`（4倍），`index=110`（`%esi`），`base=000`（`%eax`），SIB=10110000=0xB0；
5. 编码：`81 44 B0 C1 EF BE AD DE`。

- `cmpxchg %eax, 0xFACE(%edx,%edi,2)`

1. 指令格式为 `cmpxchg r32 mem`，对应 Opcode 为 `0F B1 /r`；
2. 内存寻址：Base(%edx), Index(%edi), Scale(2), Displacement(0xFACE)；
3. ModR/M：`mod=10`（32位位移），`reg=000`（源操作数`%eax`），`r/m=100`（需要SIB），ModR/M=10000100=0x84；
4. SIB：`scale=01`（2倍），`index=111`（`%edi`），`base=010`（`%edx`），SIB=01111010=0x7A；
5. 编码：`0F B1 84 7A CE FA 00 00`。

- `movl 0xCAFE(%ebx,%esi,4) %eax`

1. 指令格式为 `movl mem, r32`，对应 Opcode 为 `8B /r`；
2. 内存寻址：Base(%ebx), Index(%esi), Scale(4), Displacement(0xCAFE)；
3. ModR/M：`mod=10`（32位位移），`reg=000`（目标寄存器`%eax`），`r/m=100`（需要SIB），ModR/M=10000100=0x84；
4. SIB：`scale=10`（4倍），`index=110`（`%esi`），`base=011`（`%ebx`），SIB=10110011=0xB3；
5. 编码：`8B 84 B3 FE CA 00 00`。

x86-64 指令

- `movq 0x100(%rsp,%rbx,8), %rax`

1. REX前缀：`0x48=01001000`，`W=1`表示64位操作；
2. 指令格式为 `mov r64, r/m64`，对应 Opcode 为 `0x8B`；
3. 内存寻址：Base(%rsp), Index(%rbx), Scale(8), Displacement(0x100)；
4. ModR/M：`mod=10`（32位位移），`reg=000`（目标寄存器`%eax`），`r/m=100`（需要SIB），ModR/M=10000100=0x84；
5. SIB：`scale=11`（8倍），`index=011`（`%rbx`），`base=100`（`%rsp`），SIB=11011100=0xDC；
6. 编码：`48 8B 84 DC 00 01 00 00`。

- `addq %rax, 0x100(%rsp,%rbx,8)`

1. REX前缀：`0x48=01001000`，`W=1`表示64位操作；
2. 指令格式为 `add r/m64, r64`，对应 Opcode 为 `0x01`；
3. 内存寻址：Base(%rsp), Index(%rbx), Scale(8), Displacement(0x100)；
4. ModR/M：`mod=10`（32位位移），`reg=000`（目标寄存器`%eax`），`r/m=100`（需要SIB），ModR/M=10000100=0x84；
5. SIB：`scale=11`（8倍），`index=011`（`%rbx`），`base=100`（`%rsp`），SIB=11011100=0xDC；

6. 编码：`48 01 84 DC 00 01 00 00`。