```
#include <stdio.h>
/* Example use of switch statement */

long switchv(long idx) {
    long result = 0;
    switch(idx) {
    case 0:
        result = 0xaaa;
        break;
    case 2:
    case 5:
        result = 0xbbb;
        break;
    case 3:
        result = 0xccc;
        break;
    default:
        result = 0xddd;

    }
    return result;
}

/* Testing Code */
#define CNT 8
#define MINVAL -1

int main() {
    long vals[CNT];
    long i;
    for (i = 0; i < CNT; i++) {
        vals[i] = switchv(i + MINVAL);
        printf("idx = %ld, val = 0x%lx\n", i + MINVAL, vals[i]);
    }
    return 0;
}
```

图 4-69　Switch 语句可以翻译成 Y86-64 代码。这要求实现一个跳转表

* 4.51　练习题 4.3 介绍了 iaddq 指令，即将立即数与寄存器相加。描述实现该指令所执行的计算。参考 irmovq 和 OPq 指令的计算(图 4-18)。

** 4.52　文件 seq-full.hcl 包含 SEQ 的 HCL 描述，并将常数 IIADDQ 声明为十六进制值 C，也就是 iaddq 的指令代码。修改实现 iaddq 指令的控制逻辑块的 HCL 描述，就像练习题 4.3 和家庭作业 4.51 中描述的那样。可以参考实验资料获得如何为你的解答生成模拟器以及如何测试模拟器的指导。

*** 4.53　假设要创建一个较低成本的、基于我们为 PIPE—设计的结构(图 4-41)的流水线化的处理器，不使用旁路技术。这个设计用暂停来处理所有的数据相关，直到产生所需值的指令已经通过了写回阶段。

文件 pipe-stall.hcl 包含一个对 PIPE 的 HCL 代码的修改版，其中禁止了旁路逻辑。也就是，信号 e_valA 和 e_valB 只是简单地声明如下：

```
##  DO NOT MODIFY THE FOLLOWING CODE.
## No forwarding.  valA is either valP or value from register file
word d_valA = [
        D_icode in { ICALL, IJXX } : D_valP; # Use incremented PC
        1 : d_rvalA;  # Use value read from register file
];
```