```
define i32 main() {
 a = alloca [100 x i32]; //allocate local array
 return foo(a, 60);
}
define i32 foo(i32* a, i32 b) {
  cmp = icmp sgt i32 b, 0
  br cmp, label for.body, label for.end
for.body:
  // phi node merge index from two basic blocks
  idx = phi i64 [idx.next], [0]
  addr = getelementptr(a, idx);
  c += load addr;
  idx.next = idx + 1;
  cond = icmp eq i32 idx.next, b;
  br cond, label for.end, label for.body
for.end:
  return c;
}
```

(b) LLVM IR generated by Clang

```
State = {enum Arch, i64 Regs[], i8 Flags, ...}
define Mem main(State* s, PC pc, Mem mem){
  // prepare local variables for CPU registers
  eax = s.Regs[0]; ebx = s.Regs[1]; ...
  Mem mem1 = foo(s, pc + 12, mem);
  eax = s.Regs[0]; // load return value
  ...
  // update global state s
  s.Regs[0] = eax; s.Regs[1] = ebx; ...
  return mem1;
}
define Mem foo(State* s, PC pc, Mem mem){
 ...
 Block_4005f1:
 // load array a with offset k; k is in eax
  ebx = _read_mem(mem, esi + eax);
 // store c (in ebx) on top of the stack
  Mem mem1 = _write_mem(mem, esp, ebx);
```

(c) LLVM IR lifted by McSema

```
 block_exit:
  eax = __read_memory(mem, esi + eax);
 ...
}
define i32 _read_mem(Mem mem, i32 offset) {...}
define Mem _write_mem(Mem mem, i32 offset) {...}
```

(c) LLVM IR lifted by McSema (con'd)

```
 int main() {
   int a[100];
   int c = foo(a, 60);
   return c;
 }
 int foo(int a[], int b) {
   int k, c;
   for (k = 0; k < b; k++)
     c += a[k];
   return c;
 }
```

(a) Original C code