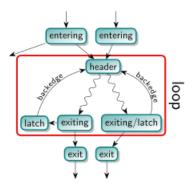
Loop in LLVM

1 Loop

I was surprised to learn that loops can be defined at the LLVM IR level. In higher-level programming languages like C, loops have their own syntax and are clearly a language structure. However, this is not the case in LLVM IR.

Despite this, LLVM IR does have a definition of a loop. A loop consists of three main parts: a header, an exiting block, and a latch.



Here is an example of loop written in LLVM IR.

```
define void @test(i32 %n) {
entry:
    br label %body
body:
    %i = phi i32 [ 0, %entry ], [ %i.next, %latch ]
    ; Loop body
    br label %latch
latch:
    %i.next = add nsw i32 %i, 1
    %cond = icmp slt i32 %i.next, %n
    br i1 %cond, label %body, label %exit
exit:
    ret void
}
```

It's CFG generated by compiler explorer.

```
loop:
        mov
                 dword ptr [rsp - 8], edi
                 eax, eax
        xor
                 dword ptr [rsp - 4], eax
        mov
                 .LBB1_1
        jmp
.LBB1_1:
                eax, dword ptr [rsp - 4]
        mov
                dword ptr [rsp - 12], eax
        mov
                ecx, dword ptr [rsp - 8]
        mov
                eax, dword ptr [rsp - 12]
        mov
        add
                eax, 1
                eax, ecx
        cmp
                dword ptr [rsp - 4], eax
        mov
        j1
                .LBB1_1
                .LBB1_1:@33
                        ret
```

2 Cycle

In LLVM, cycles are not considered loops. Instead, they are blocks that link to each other in the control flow graph (CFG) without a common header block to jump back to.



Here is a cycle:

```
define void @test(i32 %n) {
entry:
  ; Check if n is 2077
 %is2077 = icmp eq i32 %n, 2077
  ; If n is 2077, jump to latch, otherwise jump to body
 br i1 %is2077, label %latch, label %body
 %i = phi i32 [ 0, %entry ], [ %i.next, %latch ]
  ; Loop body
 br label %latch
latch:
  %i.in = phi i32 [%i, %body], [0, %entry]
 %i.next = add nsw i32 %i.in, 1
 %cond = icmp slt i32 %i.next, %n
 br i1 %cond, label %body, label %exit
exit:
 ret void
}
```

And its CFG:

```
cycle:
                                                                     dword ptr [rsp - 12], edi
                                                      mov
                                                                     eax, eax
edi, 2077
ecx, eax
dword ptr
                                                      xor
                                                      cmp
                                                      mov
                                                                     dword ptr [rsp - 8], ecx
dword ptr [rsp - 4], eax
                                                      mov
                                                      mov
                                                      je
                                                                     .LBB0 2
                                        .LBB0_2:
                                                                    ecx, dword ptr [rsp - 12]
eax, dword ptr [rsp - 4]
eax, 1
eax, ecx
dword ptr [rsp - 8], eax
.LBB0_1
                                                      mov
                                                      mov
                                                      add
                                                      cmp
                                                      mov
jl
                                                                                                             .LBB0_2:@18
.LBB0_1:
                           eax, dword ptr [rsp - 8]
dword ptr [rsp - 4], eax
             mov
             mov
```

A cycle in LLVM IR becomes a loop if all edges from outside the subset of blocks that form the cycle point to the same node, which is called the header. Conversely, a cycle is not considered a loop if not all external edges point to the header.

3 Reference

Loop Definition (https://llvm.org/docs/LoopTerminology.html)