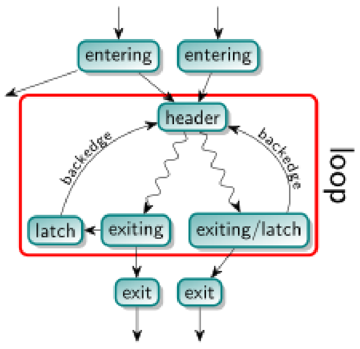
Loop in LLVM

1 Loop

It suprise me that the loop can be defined at LLVM IR level. In higher level programming language like C, loop has its own syntax thus clearly are a language structure, but it's not the case in LLVM IR.

Anyway, LLVM IR has a definition of loop. A loop has mainly three parts: header, exiting, and latch.



Here is an example of loop written in LLVM IR.

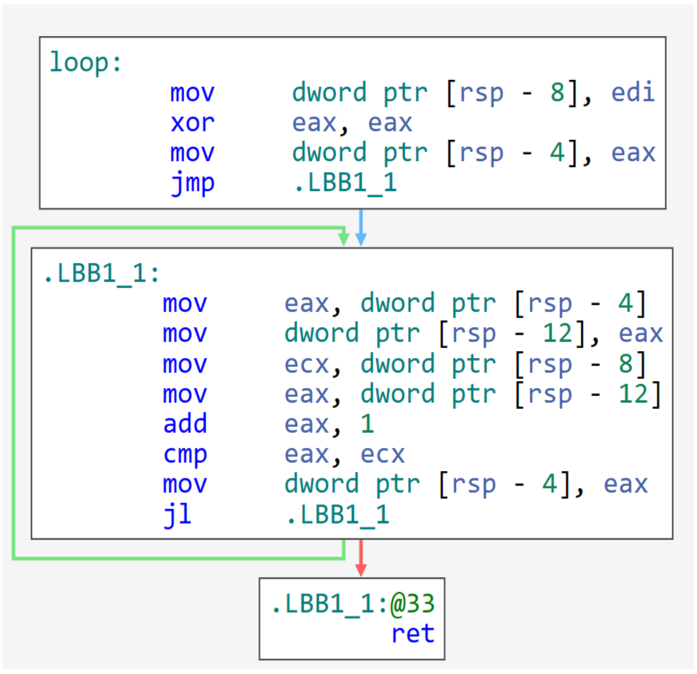
```
define void @test(i32 %n) {
entry:
  br label %body

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.



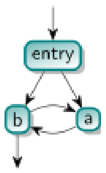
```
loop:
        mov     dword ptr [rsp - 8], edi
        xor     eax, eax
        mov     dword ptr [rsp - 4], eax
        jmp     .LBB1_1
```

```
.LBB1_1:
        mov     eax, dword ptr [rsp - 4]
        mov     dword ptr [rsp - 12], eax
        mov     ecx, dword ptr [rsp - 8]
        mov     eax, dword ptr [rsp - 12]
        add     eax, 1
        cmp     eax, ecx
        mov     dword ptr [rsp - 4], eax
        jl      .LBB1_1
```

```
.LBB1_1:@33
        ret
```

2 Cycle

Cycles are not Loops in LLVM. They are blocks linking each other (in CFG) without a common header block to jump back to.



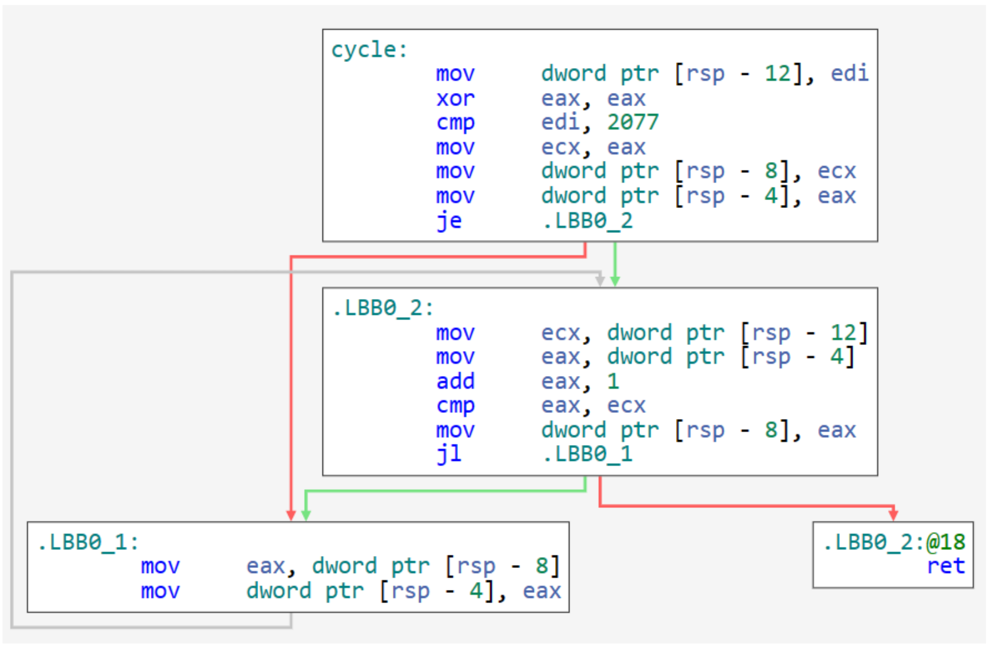
Here is a cycle:

```
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


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:
        mov     dword ptr [rsp - 12], edi
        xor     eax, eax
        cmp     edi, 2077
        mov     ecx, eax
        mov     dword ptr [rsp - 8], ecx
        mov     dword ptr [rsp - 4], eax
        je      .LBB0_2

.LBB0_2:
        mov     ecx, dword ptr [rsp - 12]
        mov     eax, dword ptr [rsp - 4]
        add     eax, 1
        cmp     eax, ecx
        mov     dword ptr [rsp - 8], eax
        jl      .LBB0_1

.LBB0_1:
        mov     eax, dword ptr [rsp - 8]
        mov     dword ptr [rsp - 4], eax

.LBB0_2:@18
        ret
```

All edges from outside the subset into the subset point to the same node, called the header. Cycle is not a loop, because not all external edges point to the header.

3 Reference

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