

Project 04

Branching - offset, conditional

Remaining Functions - ls, lcr, cmp, mov, mul

ELF lib + example (tomorrow)

MOV R0, $\boxed{\#-1}$ → mn

1111 1111 (-1)
0000 0000
+1

0000 0001 (1)

01

0000 0001
1111 -- 1111 1110 (-2)

MOV R0, # -1
-2
-3

#-16

run

Branching

index loop:

0	add	→	offset - word offset · 24 bits · 2's comp · relative to cur index + 2 (PC + 8)
1	sub	→	
2	ldr	→	
3	b	loop →	
4	add	loop-end	
5	add	loop-end	
6	sub	loop-end	
7	loop-end:	ldr	
8	str		

codegen for b loop

int index = 3

int target-index = 0

$$\text{offset} = \text{target-index} - (\text{index} + 2)$$

$$0 - (3 + 2)$$

$$0 - 5$$

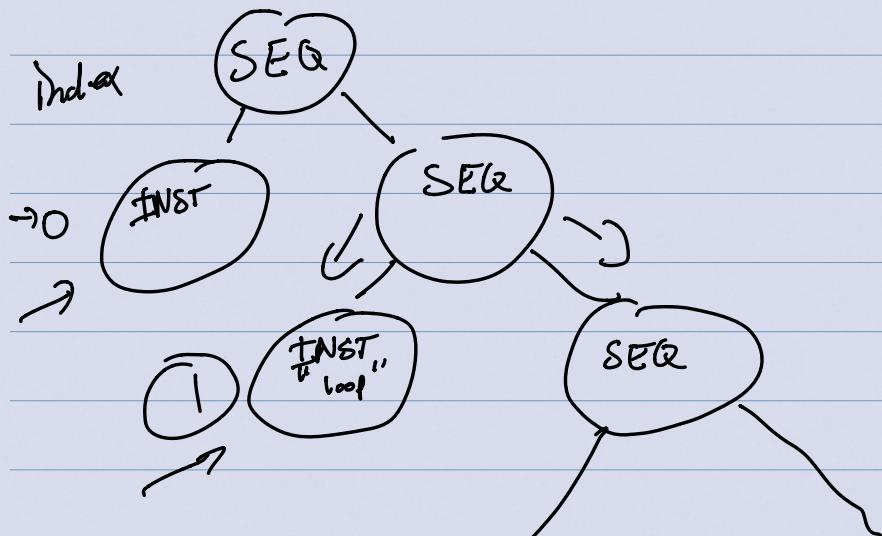
$$= \boxed{-5}$$

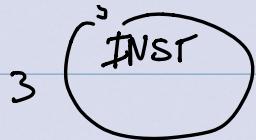
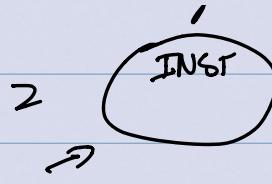
$$\begin{aligned}
 \text{int index} &= 3 \\
 \text{int target_index} &= 7 \\
 \text{offset} &= 7 - (3 + 2) \\
 &= 7 - 5 \\
 &= 2
 \end{aligned}$$



* ~~codegen-b (ct, np)~~ \in parse-tree
~~int index = ?ct->len;~~
~~int target-index = codegen-get-index(, label, o)~~

[Cond]





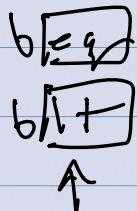
```

int codegen_get_index(struct parse_node *np, char *label, int level) {
    int l1 = -1;
    int l2 = -1;

    if (np->type == INST) {
        if (strcmp(label, np->start.inst.label, SCAN_TOKEN_LEN) == 0) {
            l1 = level;
        }
    } else if (np->type == SEQ) {
        l1 = codegen_get_index(np->start.seq.left, label, level);
        l2 = codegen_get_index(np->start.seq.right, label, level + 1);
        if (l1 == -1) {
            l1 = l2;
        }
    }
    return l1;
}

```

conditional execution



MOVEq

},

→ lsr r0, r1, #1

→ E1A0 00A1

The diagram illustrates the assembly instruction `MOV S rD` with binary values:

- Cond:** 1110
- dp:** 0001
- Mov:** 1010
- S:** 0000
- Rn:** 0000
- rd:** 0000
- rD:** 1010
- rm:** 0001

The **Cond** and **dp** fields are combined as **E**. The **Mov** field is combined with **S** and **Rn** to form the **rm** field. The **rd** and **rD** fields are combined to form the **rD** value.

A separate diagram shows the **rm** field being split into **1** and **lr**.