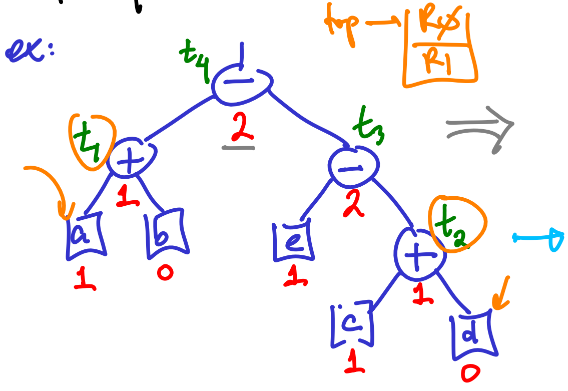
CS445 03/14/19

Chapter 9.10: Code Generation from DAGs.

ex:



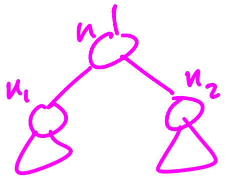
	registers		temporary word
	R∅	R1	t1
mov a, R∅	a		
add b, R∅	a+b		
mov c, R1		c	
add d, R1		c+d	
mov R∅, t1			a+b
mov e, R∅	e		
sub R1, R∅	e-(c+d)		
mov t1, R1		a+b	
sub R∅, R1		(a+b)-(e-(c+d))	
mov R1, t1			

label:

gencode

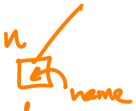
$$label(n)=\begin{cases}1 & \text{if } n \text{ is a left leaf}\\ 0 & \text{if } n \text{ is a right leaf}\\ \max\{l_1,l_2\} & \text{if } \underbrace{label(n_1)}_{l_1} \neq \underbrace{label(n_2)}_{l_2}\\ 1+l_1 & \text{if } \dots = \dots\end{cases}$$

leaf case



gencode(n):

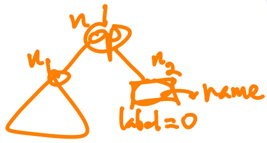
Case 0:



n is a left leaf

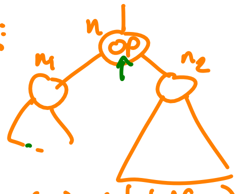
```
[print "Mov name, top(rstack)"]
```

Case 1:



```
gencode(n1)
print "OP name, top(rstack)"
```

Case 2:

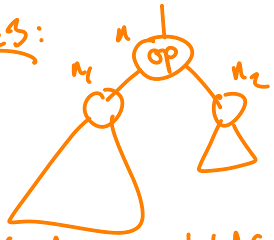


$1 \le label(n_1) < label(n_2)$

and $label(n_1) < r$

```
swap(rstack)
gencode(n2)
R ← pop(rstack)
gencode(n1)
print "OP R, top(rstack)"
push(R)
swap(rstack)
```

Case 3:



$1 \le label(n_2) \le label(n_1)$

and $label(n_2) < r$

```
gencode(n1)
R ← pop(rstack)
gencode(n2)
print "OP top(rstack), R"
push R
```

Case 4:

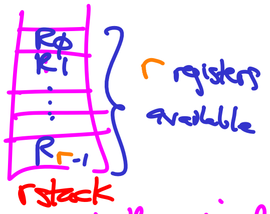
$label(n_1), label(n_2) \ge r$

Assumptions:

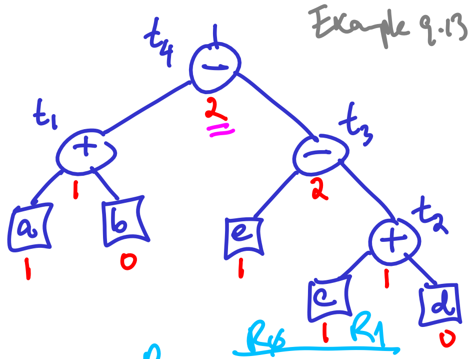
1) Register stack

2) R∅ (top of stack) will contain the value of n

3) Stack will be restored to the original configuration at the end of the algorithm



Example 9.13 : p565

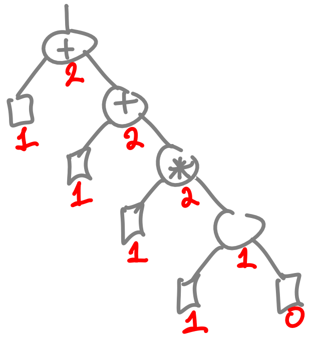


	Rφ	R1
mov e, R1		e
mov c, Rφ	c	
add d, Rφ	c+d	
sub Rφ, R1		e−(c+d)
mov a, Rφ	a	
add b, Rφ	a+b	
sub R1, Rφ	(a+b)−(e−(c+d))	

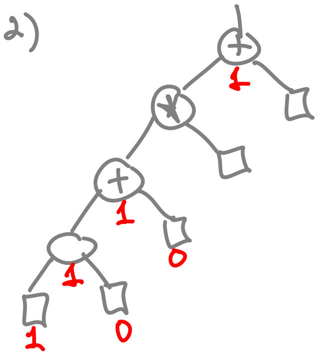
```
                          top
• gencode(t4)    [R1, Rφ]        /* Case 2 */
  • swap
  • gencode(t3)  [Rφ, R1]        /* Case 3 */

     • gencode(e)  [Rφ, R1]      /* Case φ */
         print "Mov e, R1"
     • R1 ← pop
     • gencode(t2)  [Rφ]         /* case 1 */
        • gencode(c)  [Rφ]       /* case φ */
            print "mov c, Rφ"
        • print "ADD d, Rφ"
     • print "SuB Rφ, R1" ←
     • push R1 ←
  • R1 ← pop
  • gencode(t1)  [Rφ]            /* Case 1 */
        • gencode(a)  [Rφ]       /* Case φ */
              print "mov a, Rφ"
        • print "ADD b, Rφ"
  • print "SuB R1, Rφ"
  • push R1          [Rφ, R1]
  • swap             [R1, Rφ]
```

Questions:

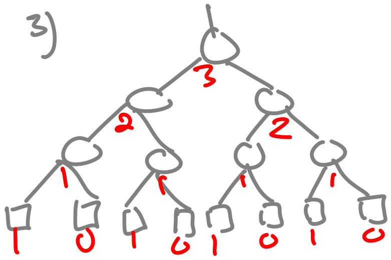
1)



2)



3)



3
2
2
1
1
1
1
1
0
1
0
1
0
1
0