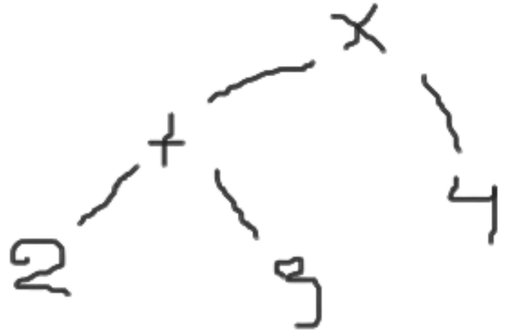
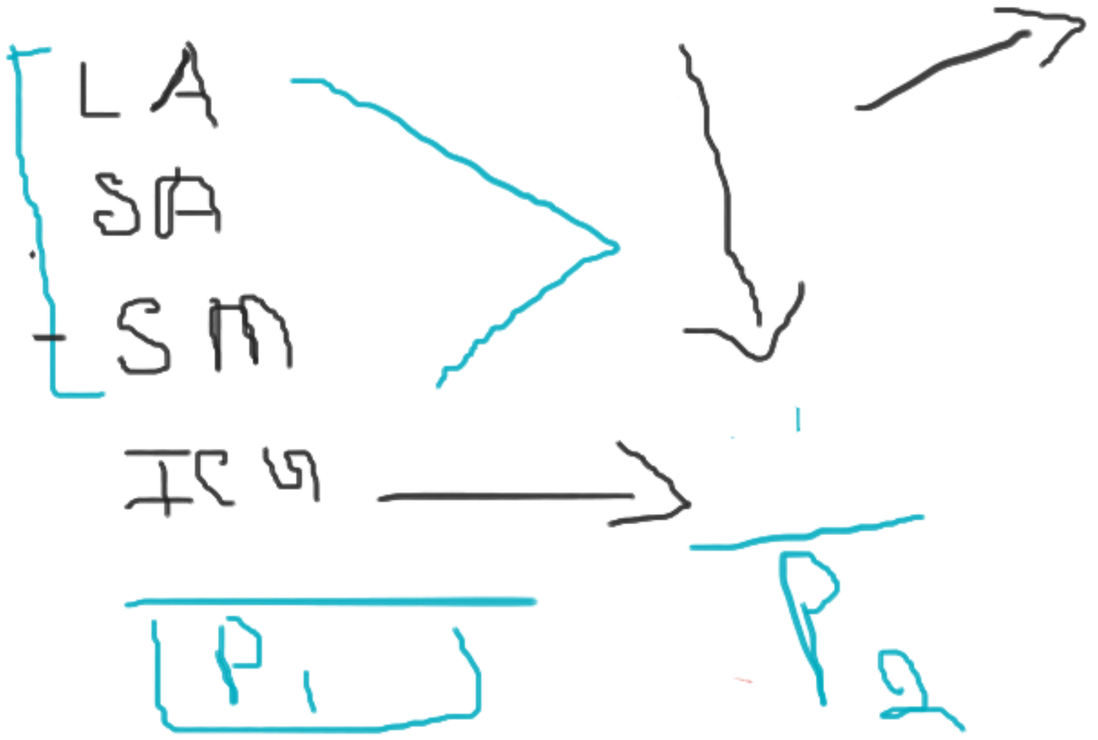
Intermediate code generation

- Machine Independent
- Abstract Syntax Tree
- Direct acyclic Graph
- postfix
- 3 address code

(2+3)*4





Types of three address statements:

Assignment x = y op z
x = op z
x = y

Jump conditional: if x then goto L
unconditional : goto L

Array assignment x = y [i]
x [i] = y

pointer, address assign x = & y
x = **y

Example 01: x = (a * b) + (c * d)

$t_1 = c \times d$

$t_2 = a \times b$

$t_3 = t_1 + t_2$

$x = t_3$

Example 2: a*-(b + c)

$t_1 = b + c$

$t_2 = \text{uminus } t_1$

$t_3 = a * t_2$

Example 3: a = b + c * d

$t_1 = c * d$

$t_2 = b + t_1$

$a = t_2$

$a[2] + = 5 * 6$

$t_1 = 5 * 6$

$t_2 = a[2]$

$a[2] = t_1 + t_2$

Quadruples

operator
arg1
arg2
result

a = b * - c + b * - c

t1 = uminus c
t2 = b * t1
t3 = uminus c
t4 = b * t3
t5 = t2 + t4
a = t5

	op	arg1	arg2	result
(0)	uminus	c	—	t1
(1)	*	b	t1	t2
(2)	uminus	c	—	t3
(3)	*	b	t3	t4
(4)	+	t2	t4	t5
(5)	=	t5		a

Triple

1. t1= uminus c
2. t2 = b * t1
3. t3 = uminus c
4. t4 = b * t3
5. t5 = t2 + t4
6. a = t5

	op	arg1	arg2
(0)	uminus	c	
(1)	*	b	(0)
(2)	uminus	c	
(3)	*	b	(2)
(4)	+	(1)	(3)
(5)	=	(4)	

(a + b) * (c + d) - (a + b + c)

t1 = a + b
t2 = c + d
t3 = t1 * t2
t4 = t1 + c
t5 = t3 - t4

Quadruple

	op	a1	a2	Result
(0)	+	a	b	t_1
(1)	+	c	d	t_2
(2)	x	t_1	t_2	t_3
(3)	+	t_1	c	t_4
(4)	—	t_2	t_4	t_5

Triple

	op	a1	a2
(0)	+	a	b
(1)	+	c	d
(2)	x	(0)	(1)
(3)	+	(0)	C
(4)	—	(2)	(3)