PES UNIVERSITY

CELEBRATING 50 YEARS

Compiler Design

Preet Kanwal

Department of Computer Science & Engineering

Teaching Assistant : Kavya P K

Compiler Design

Unit 3: Three-Address Code

Preet Kanwal

Department of Computer Science & Engineering

Lecture Overview

In this lecture, you will learn about -

- Data Structures for Three-Address Code
  - Quadruples
  - Triples
  - Indirect Triples
  - Example Questions

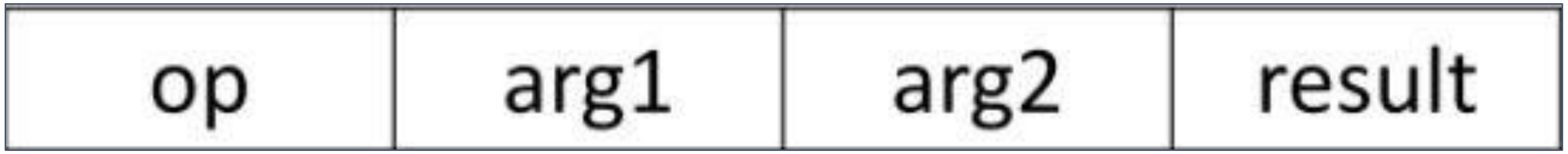
Data Structures for TAC

- Three address code is represented as a record structure with fields for operator and operands.
- These records can be stored as an array or a linked list.
- There three types of record structures -

1. Quadruples [4 fields]
2. Triples [3 fields]
3. Indirect Triples [Triples + List of pointers to Triples]

Quadruples

- A Quadruple is an array type data structure with 4 fields -



op	arg1	arg2	result

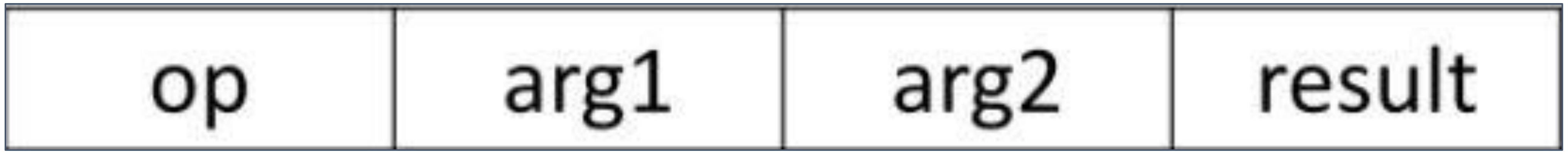
where,

op - operator.

arg1, arg2 - the two operands used.

result - the result of the expression.

Quadruples



op	arg1	arg2	result

- **arg1, arg2 and result are pointers to symbol table entries.**
- **This means even temporaries must be placed in symbol table as they are created.**
- **Any unused field is left blank/NULL**
- **Disadvantage - Temporary names have to be entered into symbol table.**

Quadruples Format - Unary Operators

The given table describes the quadruple format for unary operators -

Statement	op	arg1	arg2	result
Unary operators - arg2 is empty	op	arg1	null	arg2
Example: x=-y	-	y	null	x
Example: x=y	=	y	null	x

Quadruples Format - Functions

The given table describes the quadruple format for functions -

Statement	op	arg1	arg2	result
param operator - arg2 and result are empty	param	arg1	null	null
Example: param x	param	x	null	null
Function Call - call func_name, func_param	call	func_name	value	x
Example: call foo,3	call	foo	3	null
Example: x = call foo,3	call	foo	3	x

Quadruples Format - Jumps

The given table describes the quadruple format for jumps -

Statement	op	arg1	arg2	result
For unconditional jumps - result is label	goto	null	null	label
conditional jump Example - if x goto L	if	x	null	L
conditional jump Example - ifFalse x goto L	ifFalse	x	null	L

Quadruples Format - Labels and return

The given table describes the quadruple format for labels and return statements -

Statement	op	arg1	arg2	result
Label generation Example - L1:	Label	null	null	L1
return	return	null	null	null
return x	return	x	null	null

Quadruples Format - Array indexing

The given table describes the quadruple format for array indexing -

Statement	op	arg1	arg2	result
x[i] = y	[]=	x	i	y
	STAR	x	i	y
x = y[i]	=[]	y	i	x
	LDAR	y	i	x

Exercise 1

Write the Three-Address Code and corresponding Quadruple representation for the following code snippet -

```
if x == 0
        u = 1;
else
    u = fact(x – 1) * x;
value = u;
```

Exercise 1 - Solution

Three-Address Code -

```
if x == 0
        u = 1;
else
    u = fact(x – 1) * x;
```

```
t1 = x == 0
ifFalse t1 goto L1

u = 1

goto L2

L1:

t2 = x – 1

param t2

t3 = call fact, 1

t4 = t3 * x

u = t4

L2:
```

Exercise 1 - Solution

Quadruple -

t1 = x == 0

ifFalse t1 goto L1

u = 1

goto L2

L1:

t2 = x – 1

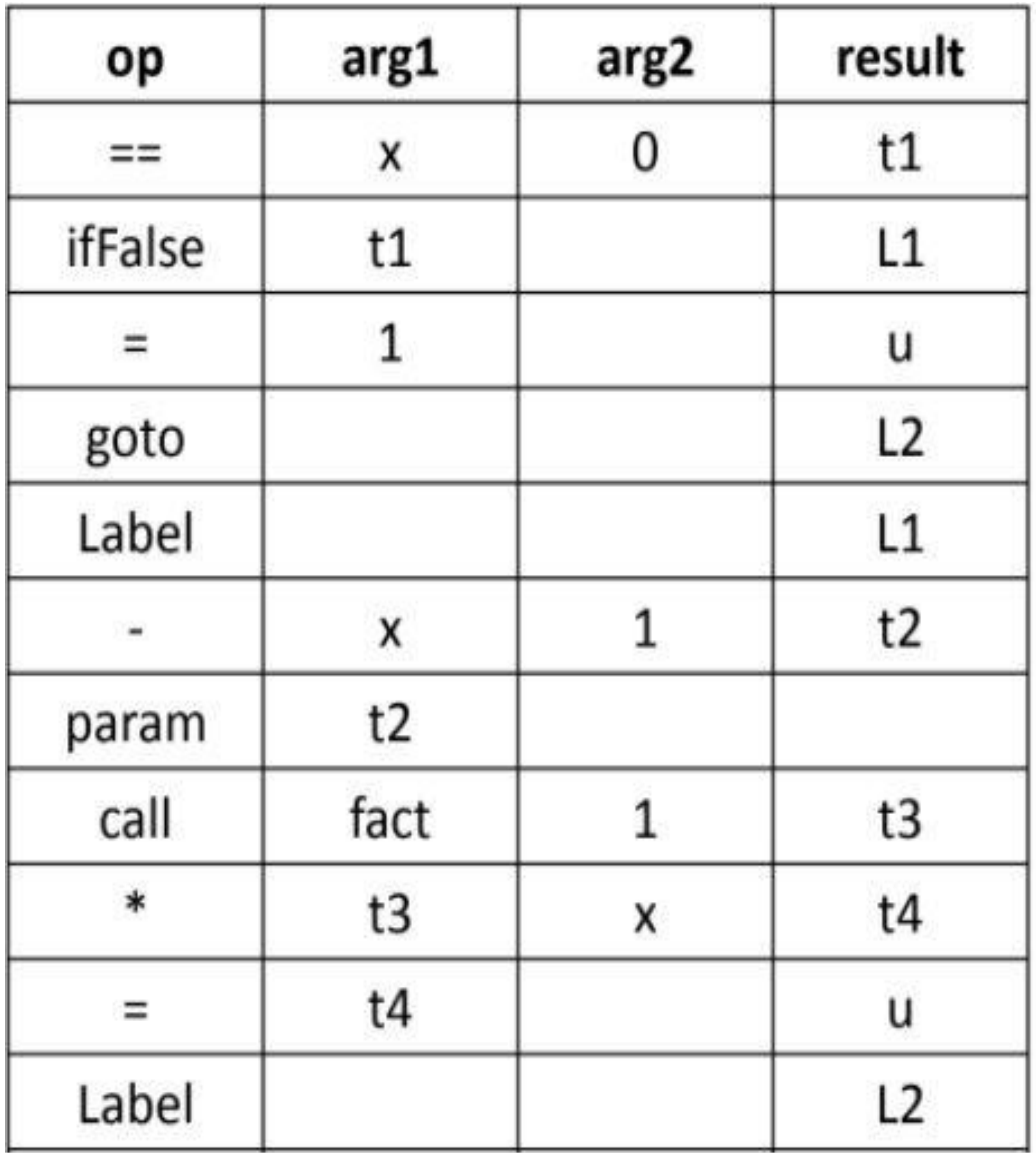
param t2

t3 = call fact, 1

t4 = t3 * x

u = t4

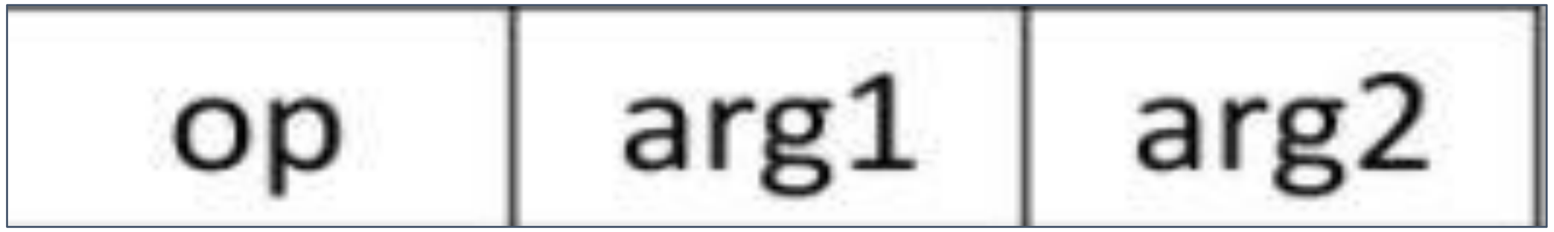
L2:



op	arg1	arg2	result
==	x	0	t1
ifFalse	t1		L1
=	1		u
goto			L2
Label			L1
-	x	1	t2
param	t2		
call	fact	1	t3
*	t3	x	t4
=	t4		u
Label			L2

Triples

- A Triple is an array type data structure with 3 fields -



op	arg1	arg2

where,

op - operator.

arg1, **arg2** - the two operands used.

Triples

- Triples are alternative ways for representing syntax tree or Directed acyclic graph.
- Triples avoid entering temporary names into symbol table.
- For a temporary, use serial number of statement computing its value.
- Problem: Code Immovability
  - No temporary variables stored in symbol table
  - All references are only to the position of statement and not location.
  - This requires the compiler to change all references to arg1 and arg2.
  - Thus, triples are not very efficient in optimizing compilers.

Triples Format - Jumps and Label

The given table describes the triple format for jumps and label -

Statement	op	arg1	arg2
Unconditional jumps	goto	(2)	
conditional jump Example - if x goto L	if	x	(2)
conditional jump Example - ifFalse x goto L	ifFalse	x	(2)
Label	Label		

Triples Format - Array indexing

The given table describes the triple format for array indexing -

Statement	Stmt no.	op	arg1	arg2
x[i] = y	(0)	[]=	x	i
	(1)	=	(0)	y
x = y[i]	(0)	=[]	y	i
	(1)	=	x	(0)

Exercise 2

Write the Triple representation for the following Three-Address Code -

```
t1 = -b

t2 = d * t1

t3 = c + t2

t4 = -b

t5 = d * t4

t6 = t3 +t5

a = t6
```

Exercise 2 - Solution

t1 = -b

t2 = d * t1

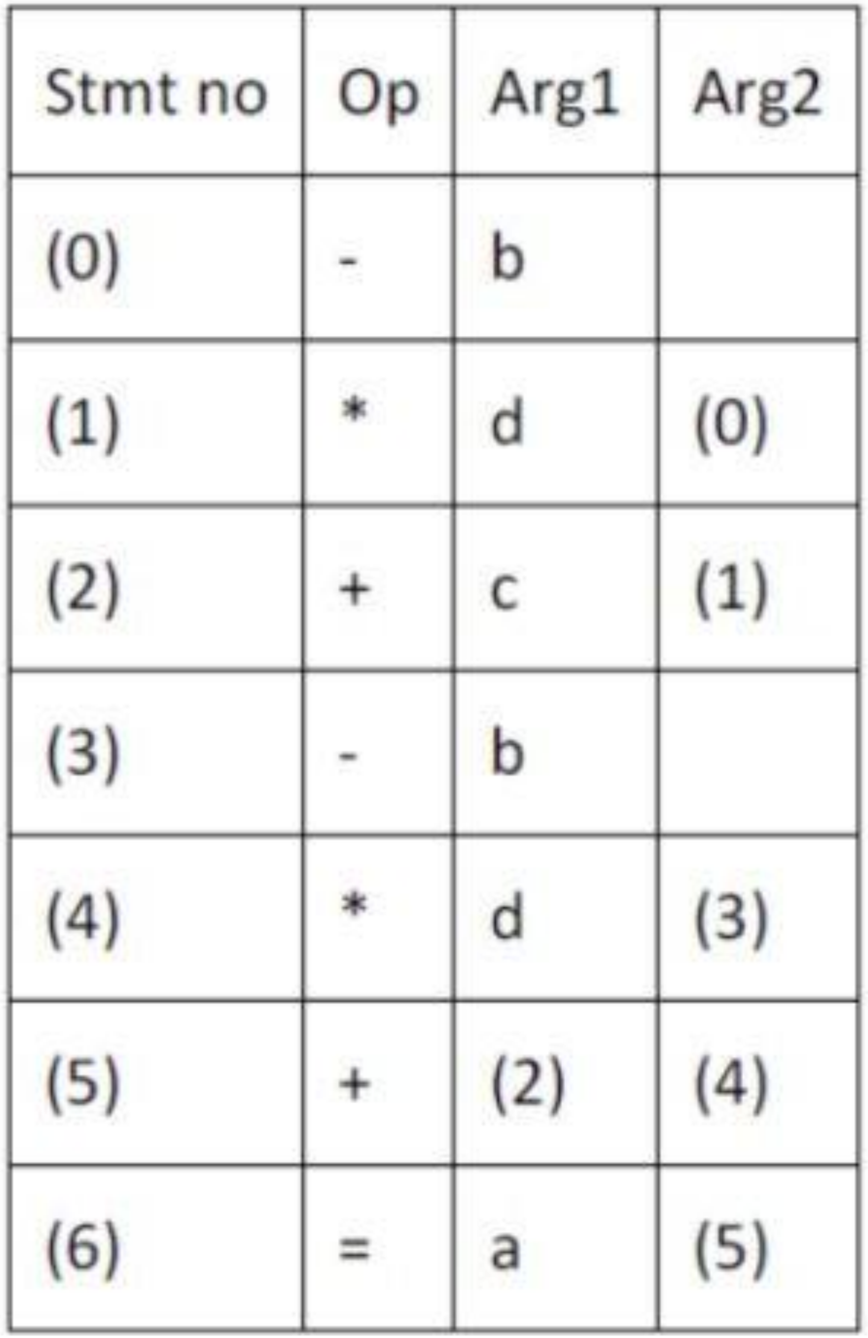
t3 = c + t2

t4 = -b

t5 = d * t4

t6 = t3 +t5

a = t6



Stmt no	Op	Arg1	Arg2
(0)	-	b	
(1)	*	d	(0)
(2)	+	c	(1)
(3)	-	b	
(4)	*	d	(3)
(5)	+	(2)	(4)
(6)	=	a	(5)

The value of a temporary variable can be accessed by the position of the statement that computes it.

Indirect Triples

- A separate list of pointers to the triple structure (i.e, statement numbers) is maintained.
- The statements can be moved by reordering the statement list.
- The utility of indirect triples is almost the same as that of quadruples, but requires less space.

Exercise 2 (cont.)

Write the Indirect Triple representation for the following Three-Address Code -

```
t1 = -b

t2 = d * t1

t3 = c + t2

t4 = -b

t5 = d * t4

t6 = t3 +t5

a = t6
```

Exercise 2 (cont.)

t1 = -b

t2 = d * t1

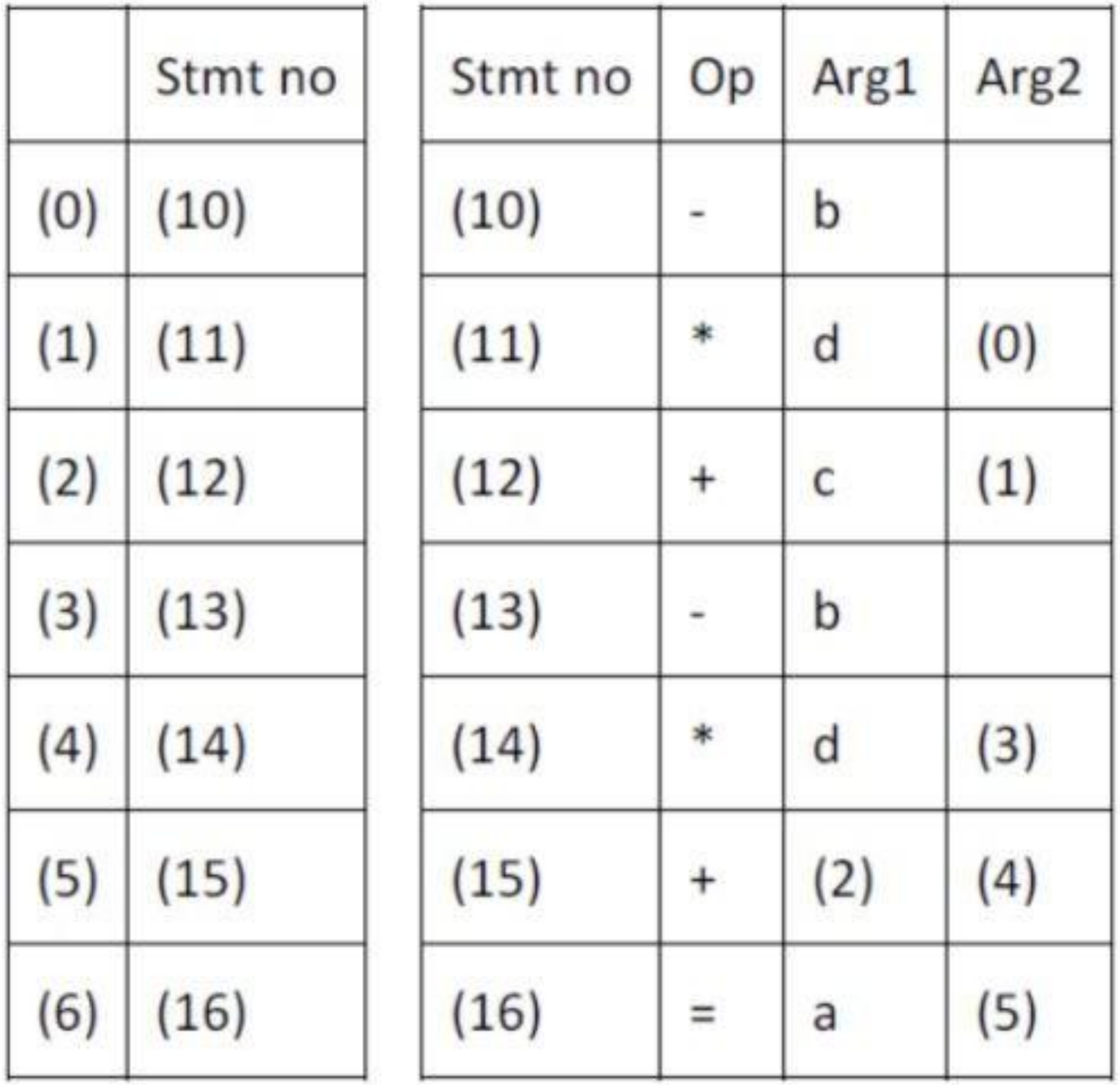
t3 = c + t2

t4 = -b

t5 = d * t4

t6 = t3 +t5

a = t6



	Stmt no
(0)	(10)
(1)	(11)
(2)	(12)
(3)	(13)
(4)	(14)
(5)	(15)
(6)	(16)

Stmt no	Op	Arg1	Arg2
(10)	-	b	
(11)	*	d	(0)
(12)	+	c	(1)
(13)	-	b	
(14)	*	d	(3)
(15)	+	(2)	(4)
(16)	=	a	(5)

No change in the Structure

Advantage of Indirect Triples

Suppose the code changes to -

t1 = -b

t2 = d * t1

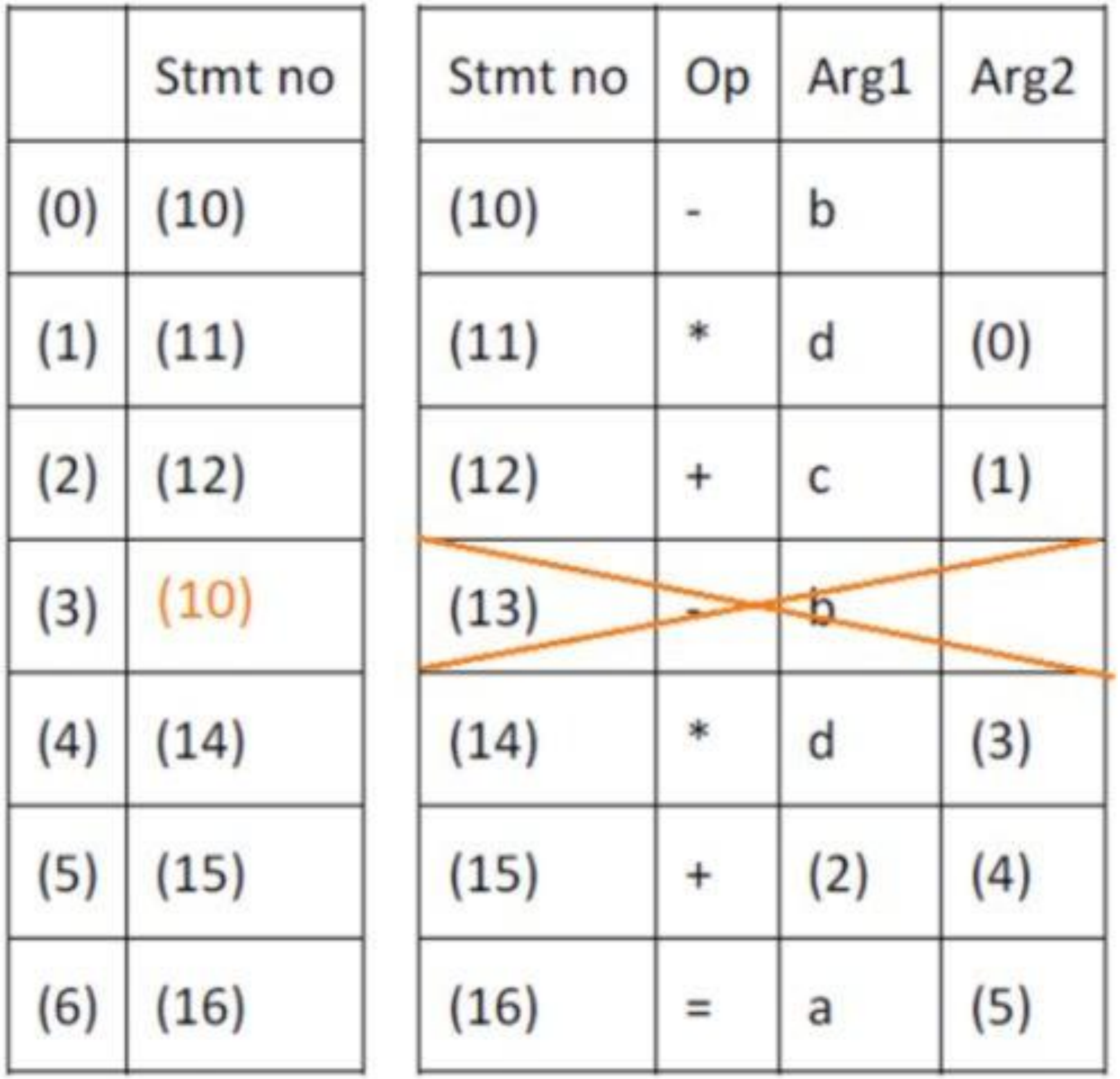
t3 = c + t2

t4 = -b

t5 = d * t4

t6 = t3 +t5

a = t6



	Stmt no
(0)	(10)
(1)	(11)
(2)	(12)
(3)	(10)
(4)	(14)
(5)	(15)
(6)	(16)

Stmt no	Op	Arg1	Arg2
(10)	-	b	
(11)	*	d	(0)
(12)	+	c	(1)
~~(13)~~	~~-~~	~~b~~	
(14)	*	d	(3)
(15)	+	(2)	(4)
(16)	=	a	(5)

No change in the Structure

Compiler Design

Exercise 3

Write the Quadruple and Triple representation for the following code snippets -

1) a = b[i] + c[j]
2) x = f(y + 1) + 2
3) X[i] = a * c + y[i] – n[j] / v
4) for(j=0; j<=10; j++)

```
{
a = a * (j* (b/c));
}
```

1) a = b[i] + c[j]

Intermediate Code -

t1 = 4 * i

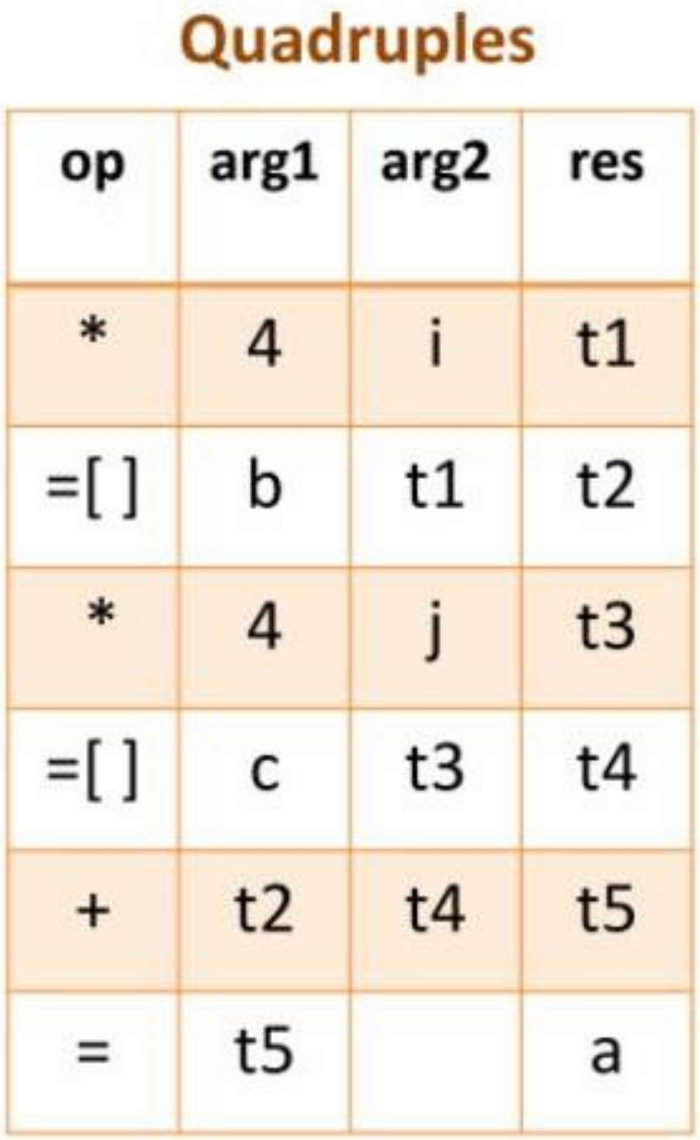
t2 = b[t1]

t3 = 4 * j

t4 = c[t3]

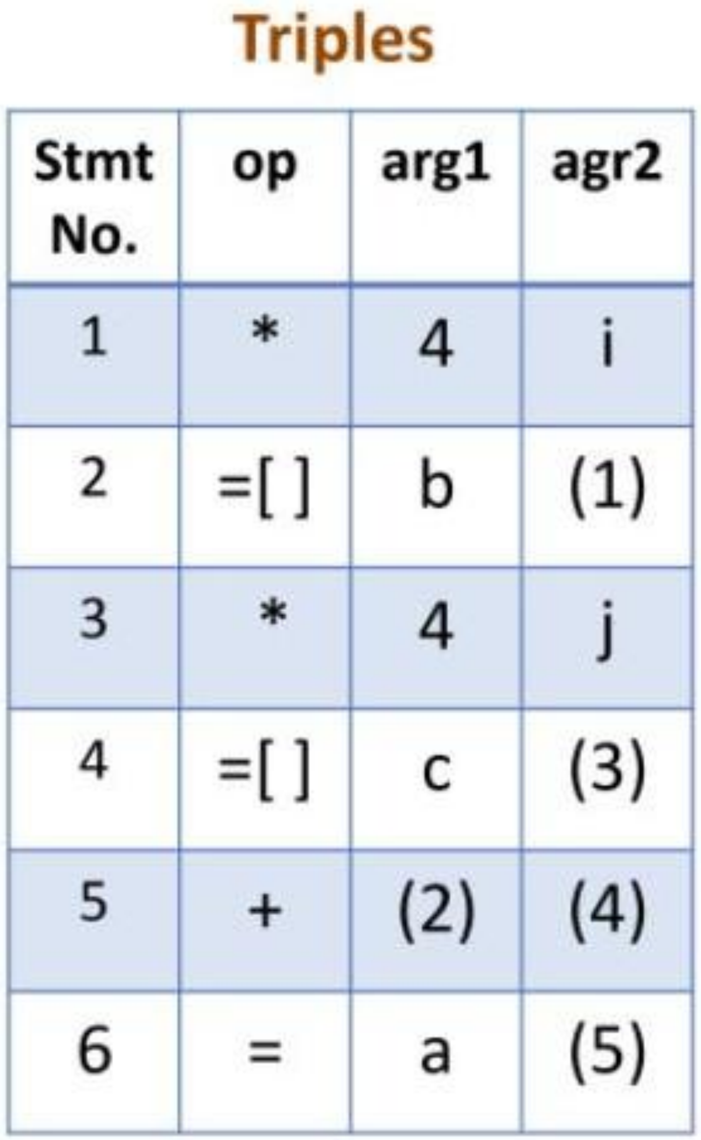
t5 = t2 + t4

a = t5



Quadruples

op	arg1	arg2	res
*	4	i	t1
=[ ]	b	t1	t2
*	4	j	t3
=[ ]	c	t3	t4
+	t2	t4	t5
=	t5		a



Triples

Stmt No.	op	arg1	agr2
1	*	4	i
2	=[ ]	b	(1)
3	*	4	j
4	=[ ]	c	(3)
5	+	(2)	(4)
6	=	a	(5)

Compiler Design

Exercise 3 - Solutions

2) x = f(y + 1) + 2

3-addr stmt	Quadruple Format						Triple Format				Indirect Triple Format						
	op	arg1	arg2	result		Stmt#	op	arg1	arg2		Ptr	Stmt#		Stmt#	op	arg1	arg2
T1 = y + 1	+	y	1	T1		1	+	y	1		11	1		1	+	y	1
Param T1	Param	T1				2	param	(1)			12	2		2	param	<11>	
T2 = call f, 1	call	f	1	T2		3	call	f	1		13	3		3	call	f	1
T3 = T2 + 2	+	T2	2	T3		4	+	(3)	2		14	4		4	+	<13>	2
X = T3	=	T3		X		5	=	X	(4)		15	5		5	=	X	<14>

3) X[i] = a * c + y[i] – n[j] / v

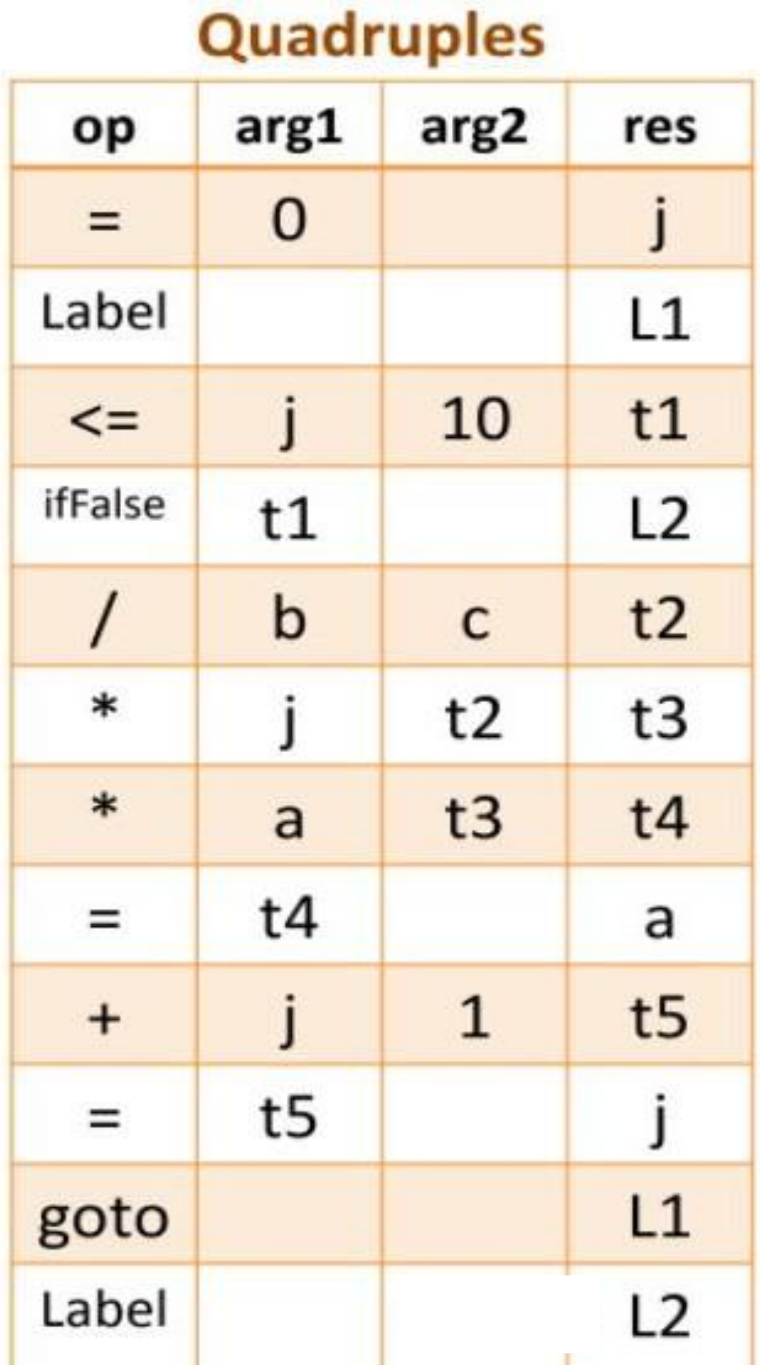
	Quadruple Format						Triple Format				Indirect Triple Format						
3-addr stmt	**op**	**arg1**	**arg2**	**result**		**Stmt #**	**op**	**arg1**	**arg2**		**Ptr**	**Stmt#**		**Stmt#**	**op**	**arg1**	**arg2**
T1 = a * c	*	a	c	T1		1	*	a	c		111	1		1	*	a	c
T2= 4 * I	*	4	I	T2		2	*	4	I		112	2		2	*	4	I
T3 = y[T2]	=[ ]	y	T2	T3		3	=[ ]	y	(2)		113	3		3	=[ ]	y	<112>
T4 = T1 + T3	+	T1	T3	T4		4	+	(1)	(3)		114	4		4	+	(1)	<113>
T5 = 4 * j	*	4	j	T5		5	*	4	j		115	5		5	*	4	j
T6 = n[T5]	=[ ]	n	T5	T6		6	=[ ]	n	(5)		116	6		6	=[ ]	n	<115>
T7 = T6/v	/	T6	v	T7		7	/	(6)	v		117	7		7	/	<116>	v
T8 = T4 - T7	-	T4	T7	T8		8	-	(4)	(7)		118	8		8	-	<114>	<117>
T9 = 4 * i	*	4	I	T9		9	*	4	I		119	9		9	*	4	I
X[T9] = T8	[ ]=	X	T9	T8		10	[ ]=	X	(9)		120	10		10	[ ]=	X	<119>
						11	=	(10)	(8)		121	11		11	=	<120>	<118>

Compiler Design

Exercise 3 - Solutions

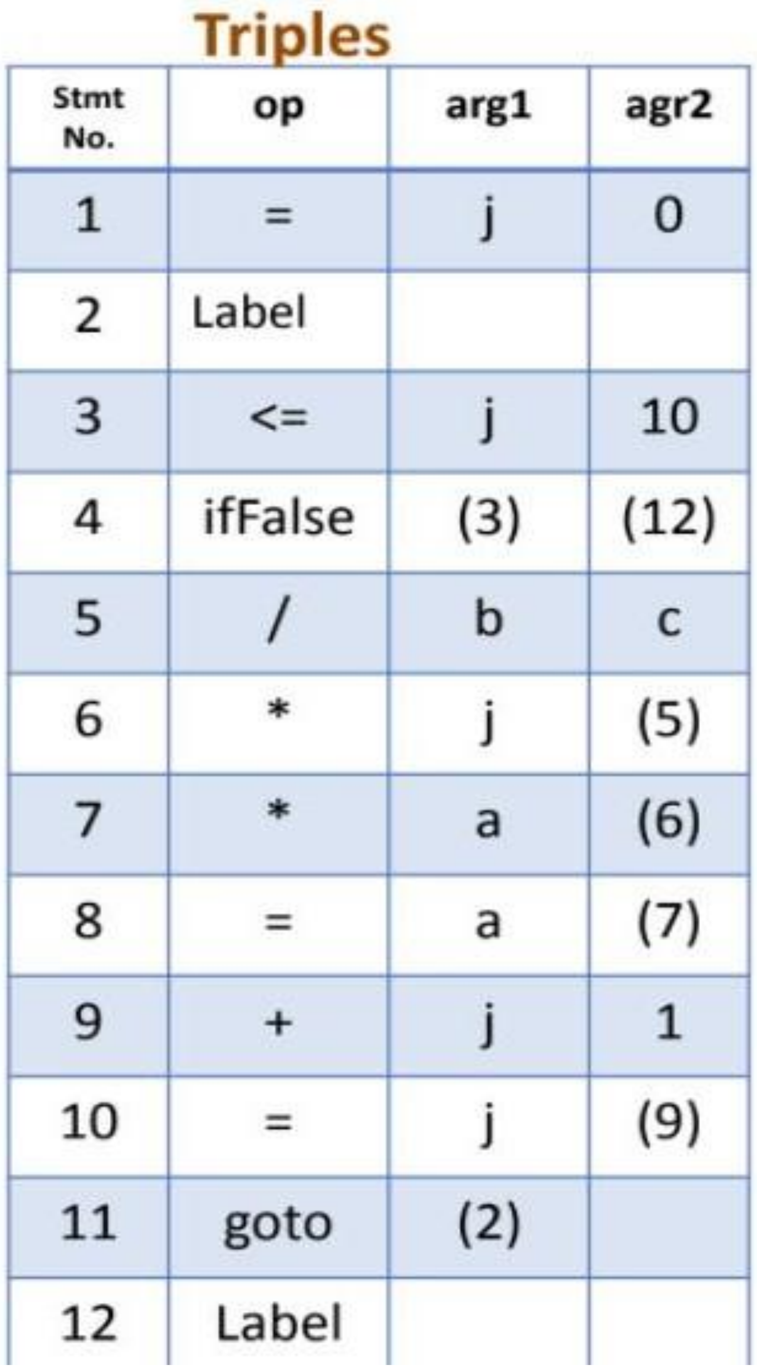
4) for(j=0; j<=10; j++){ a = a * (j* (b/c));}

```
j = 0
L1:
t1 = j <= 10
ifFalse t1 goto L2
t2 = b / c
t3 = j * t2
t4 = a * t3
a = t4
t5 = j + 1
j = t5 goto L1
L2 :
```



Quadruples

op	arg1	arg2	res
=	0		j
Label			L1
<=	j	10	t1
ifFalse	t1		L2
/	b	c	t2
*	j	t2	t3
*	a	t3	t4
=	t4		a
+	j	1	t5
=	t5		j
goto			L1
Label			L2



Triples

Stmt No.	op	arg1	agr2
1	=	j	0
2	Label		
3	<=	j	10
4	ifFalse	(3)	(12)
5	/	b	c
6	*	j	(5)
7	*	a	(6)
8	=	a	(7)
9	+	j	1
10	=	j	(9)
11	goto	(2)	
12	Label		

PES UNIVERSITY

CELEBRATING 50 YEARS

THANK YOU

Preet Kanwal

Department of Computer Science & Engineering

preetkanwal@pes.edu