FIGURE 5.16 Intermediate code for the program from Fig. 5.1.

	Operation	Op1	Op2	Result	
(1)	:=	#0		SUM	{ SUM := 0 }
(2)	:=	#0		SUMSQ	{ SUMSQ := 0 }
(3)	:=	#1		I	{ FOR I := 1 TO 100 }
(4)	JGT	I	#100	(15)	
(5)	CALL	XREAD			{ READ(VALUE) }
(6)	PARAM	VALUE			
(7)	+	SUM	VALUE	i_1	{ SUM := SUM + VALUE }
(8)	:=	i_1		SUM	

(9)	*	VALUE	VALUE	i_2	{ SUMSQ := SUMSQ + VALUE * VALUE }
(10)	+	SUMSQ	i_2	i_3	
(11)	:=	i_3		SUMSQ	
(12)	+	I	#1	i_4	{ end of FOR loop }
(13)	:=	i_4		I	
(14)	J			(4)	
(15)	DIV	SUM	#100	i_5	{ MEAN := SUM DIV 100 }
(16)	:=	i_5		MEAN	

	Operation	Op1	Op2	Result	
(17)	DIV	SUMSQ	#100	i_6	{ VARIANCE := SUMSQ DIV 100 – MEAN * MEAN }
(18)	*	MEAN	MEAN	i_7	
(19)	-	i_6	i_7	i_8	
(20)	:=	i_8		VARIANCE	
(21)	CALL	XWRITE			{ WRITE(MEAN,VARIANCE) }
(22)	PARAM	MEAN			
(23)	PARAM	VARIANCE			

FIGURE 5.16 Intermediate code for the program from Fig. 5.1. （Con't)

DIV	SUMSQ	#100	i1
*	MEAN	MEAN	i2
-	i1	i2	i3
:=	i3		VARIANCE

LDA	SUMSQ
DIV	#100
STA	T1
LDA	MEAN
MUL	MEAN
STA	T2
LDA	T1
SUB	T2
STA	VARIANCE

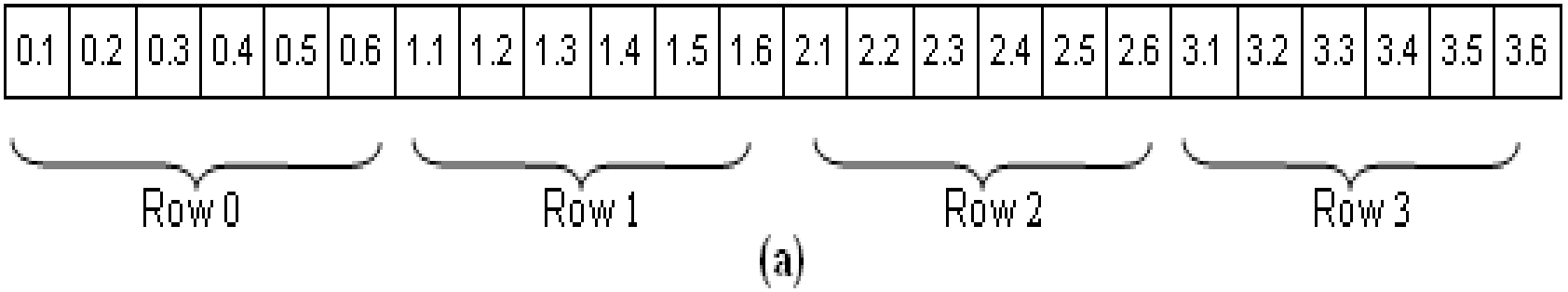
FIGURE 5.18 Rearrangement of quadruples for code optimization **(a)**

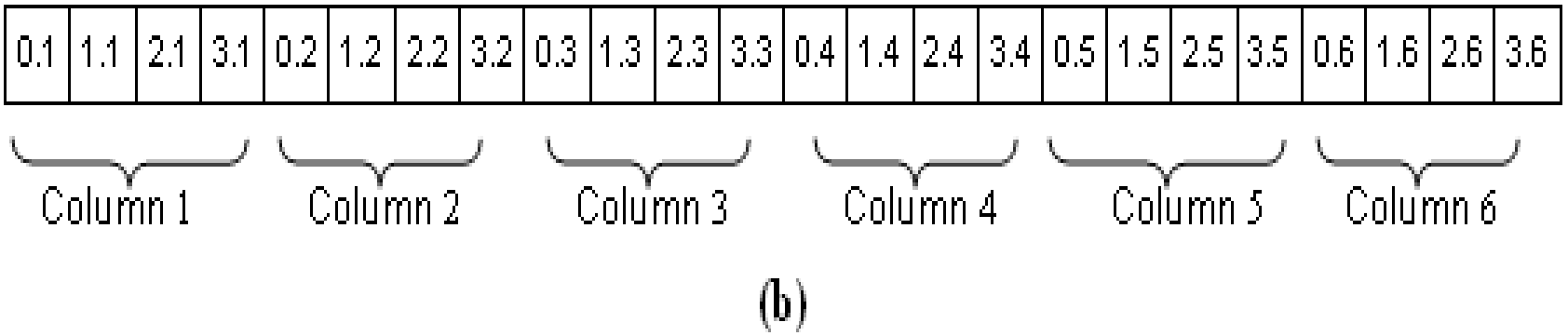
*	MEAN	MEAN	i2
DIV	SUMSQ	#100	i1
-	i1	i2	i3
:=	i3		VARIANCE

LDA	MEAN
MUL	MEAN
STA	T1
LDA	SUMSQ
DIV	#100
SUB	T1
STA	VARIANCE

FIGURE 5.18 Rearrangement of quadruples for code optimization **(b)**

FIGURE 5.21 Storage of B : ARRAY[0..3, 1..6] in (a) row-major order and (b) column-major order.





A : ARRAY [1..10] OF INTEGER
●   ●
A [I] := 5

(1)	-	I	#1	i1
(2)	*	i1	#3	i2
(3)	:=	#5		A[i2]

(a)

FIGURE 5.22 Code generation for array references.

B: ARRAY [0..3, 1..6] OF INTEGER
⋮
B [I,J] := 5

(1)	*	I	#6	i1
(2)	-	J	#1	i2
(3)	+	i1	i2	i3
(4)	*	i3	#3	i4
(5)	:=	#5		B[i4]

(b)

FIGURE 5.22 cont'd

```
X,Y : ARRAY [1..10, 1..10] OF INTEGER
    •
    •
    •
FOR  I := 1  TO  10  DO
     X [I, 2*J – 1] := Y [I, 2*J]
```

FIGURE 5.23 Code optimization by elimination of common subexpressions and removal of loop invariants.

(1)	:=	#1		I	{ loop initialization }
(2)	JGT	I	#10	(20)	
(3)	-	I	#1	i_1	{ subscript calculation for X }
(4)	*	i_1	#10	i_2	
(5)	*	#2	J	i_3	
(6)	-	i_3	#1	i_4	
(7)	-	i_4	#1	i_5	
(8)	+	i_2	i_5	i_6	
(9)	*	i_6	#3	i_7	
(10)	-	I	#1	i_8	{ subscript calculation for Y }
(11)	*	i_8	#10	i_9	
(12)	*	#2	J	i_{10}	
(13)	-	i_{10}	#1	i_{11}	
(14)	+	i_9	i_{11}	i_{12}	

(b)

FIGURE 5.23 (cont'd)

(15)	*	i_{12}	#3	i_{13}	
(16)	:=	$Y[i_{13}]$		$X[i_7]$	{ assignment operation }
(17)	+	#1	I	i_{14}	{ end of loop }
(18)	:=	i_{14}		I	
(19)	J			(2)	
(20)					{ next statement }

續 (b)

FIGURE 5.23 (cont'd)

(1)	:=	#1		I	{ loop initialization }
(2)	JGT	I	#10	(16)	
(3)	-	I	#1	i_1	{ subscript calculation for X }
(4)	*	i_1	#10	i_2	
(5)	*	#2	J	i_3	
(6)	-	i_3	#1	i_4	
(7)	-	i_4	#1	i_5	
(8)	+	i_2	i_5	i_6	
(9)	*	i_6	#3	i_7	

(10)	+	i_2	i_4	i_{12}	{ subscript calculation for Y }
(11)	*	i_{12}	#3	i_{13}	
(12)	:=	$Y[i_{13}]$		$X[i_7]$	{ assignment operation }
(13)	+	#1	I	i_{14}	{ end of loop }
(14)	:=	i_{14}		I	
(15)	J			(2)	
(16)					{ next statement }

(c)

FIGURE 5.23 (cont'd)

(1)	*	#2	J	i_3	{ computation of invariants }
(2)	-	i_3	#1	i_4	
(3)	-	i_4	#1	i_5	
(4)	:=	#1		I	{ loop initialization }
(5)	JGT	I	#10	(16)	
(6)	-	I	#1	i_1	{ subscript calculation for X }
(7)	*	i_1	#10	i_2	
(8)	+	i_2	i_5	i_6	
(9)	*	i_6	#3	i_7	

(10)	+	i_2	i_4	i_{12}	{ subscript calculation for Y }
(11)	*	i_{12}	#3	i_{13}	
(12)	:=	$Y[i_{13}]$		$X[i_7]$	{ assignment operation }
(13)	+	#1	I	i_{14}	{ end of loop }
(14)	:=	i_{14}		I	
(15)	J			(5)	
(16)					{ next statement }

(d)

FIGURE 5.23 (cont'd)

```
DO 10  I  =  1,20
    10   TABLE (I)  =  2**I
```

(a)

(1)	:=	#1		I	{ loop initialization }
(2)	EXP	#2	I	i_1	{ calculation of 2**I }
(3)	-	I	#1	i_2	{ subscript calculation }
(4)	*	i_2	#3	i_3	
(5)	:=	i_1		TABLE[i_3]	{ assignment operation }
(6)	+	I	#1	i_4	{ end of loop }
(7)	:=	i_4		I	
(8)	JLE	I	#20	(2)	

(b)

FIGURE 5.24 Code optimization by reduction in strength of operations.

(1)	:=	#1		i_1	{ initialization of temporaries }
(2)	:=	#(-3)		i_3	
(3)	:=	#1		I	{ loop initialization }
(4)	*	i_1	#2	i_1	{ calculation of 2**I }
(5)	+	i_3	#3	i_3	{ subscript calculation }
(6)	:=	i_1		TABLE[i_3]	{ assignment operation }
(7)	+	I	#1	i_4	{ end of loop }
(8)	:=	i_4		I	
(9)	JLE	I	#20	(4)	

(c)

FIGURE 5.24 (cont'd)