SEAN CONNOR
30 July 2018

Compiler Optimize Exercise - Summer 2018

Given this code fragment in an Intermediate file of a Compiler:

```
(1)   :=    #1          Indx
(2)   BGT   Indx  #25   (20)
(3)   -     Indx  #1    i1
(4)   *     i1    #10   i2
(5)   *     #3    ABC   i3
(6)   -     i3    #1    i4
(7)   -     i4    #1    i5
(8)   +     i2    i5    i6
(9)   *     i6    #4    i7
(10)  -     Indx  #1    i8
(11)  *     i8    #10   i9
(12)  *     #3    ABC   i10
(13)  -     i10   #1    i11
(14)  +     i9    i11   i12
(15)  *     i12   #4    i13
(16)  :=    Y[i13]      X[i7]
(17)  +     #1    Indx  i14
(18)  :=    i14         Indx
(19)  JMP               (2)
(20)
```

Optimize the code.
Indicate, by line number, which quadruples need to be moved, modified, or deleted.
[50 points]

Some of the possible optimization techniques:

1. Move loop invariant calculations outside the loop
2. Remove duplicate common sub-expressions
3. Reduction in Strength
4. Loop unrolling

Not all of the above possible methods are needed .

ORIGINAL						
No.	**Op**	**Source 1**	**Source 2**	**Destination**	**Description**	**Action**
(1)	:=	#1		Indx	Indx = 1	
(2)	BGT	Indx	#25	(20)	If Indx > 25 → (20)	
(3)	-	Indx	#1	i1	i1 = Indx – 1	
(4)	*	i1	#10	i2	i2 = Indx * 10	
(5)	*	#3	ABC	i3	i3 = 3 * ABC	
(6)	-	i3	#1	i4	i4 = i3 – 1	Combine to become: - i3 #2 i5
(7)	-	i4	#1	i5	i5 = i4 – 1	
(8)	+	i2	i5	i6	i6 = i2 + i5	
(9)	*	i6	#4	i7	i7 = i6 * 4	
(10)	-	Indx	#1	i8	i8 = Indx – 1	Delete because duplicate of (3)
(11)	*	i8	#10	i9	i9 = i8 * 10	Change to: * i1 #10 i9 because (10) removed
(12)	*	#3	ABC	i10	i10 = 3 * ABC	Delete because duplicate of (5)
(13)	-	i10	#1	i11	i11 = i10 – 1	Change to: + i5 #1 i11
(14)	+	i9	i11	i12	i12 = i9 + i11	
(15)	*	i12	#4	i13	i13 = i12 * 4	
(16)	:=	Y[i13]		X[i7]	X[i7] = Y[i13]	
(17)	+	#1	Indx	i14	i14 = 1 + Indx	
(18)	:=	i14		Indx	Indx = i14	
(19)	JMP			(2)	Jump → (2)	
(20)						
OPTIMIZED						
No.	**Op**	**Source 1**	**Source 2**	**Destination**	**Description**	
(1)	:=	#1		Indx	Indx = 1	
(2)	BGT	Indx	#25	(20)	If Indx > 25 → (20)	
(3)	-	Indx	#1	i1	i1 = Indx – 1	
(4)	*	i1	#10	i2	i2 = Indx * 10	
(5)	*	#3	ABC	i3	i3 = 3 * ABC	
(6)	-	i3	#2	i5	i5 = i3 – 2	
(7)	+	i2	i5	i6	i6 = i2 + i5	
(8)	*	i6	#4	i7	i7 = i6 * 4	
(9)	*	i1	#10	i9	i9 = i1 * 10	
(10)	+	i5	#1	i11	i11 = i5 + 1	
(11)	+	i9	i11	i12	i12 = i9 + i11	
(12)	*	i12	#4	i13	i13 = i12 * 4	
(13)	:=	Y[i13]		X[i7]	X[i7] = Y[i13]	
(14)	+	#1	Indx	i14	i14 = 1 + Indx	
(15)	:=	i14		Indx	Indx = i14	
(16)	JMP			(2)	Jump → (2)	
(17)						

Quadruples

A quadruples example: `sum := sum + value`

Operation	Operand	Operand	Result
+	sum	value	i1
:=	i1		sum

Array Element Address Calculations

Array address calculation method for row-major order:

Row 0 Row 1 Row 2 Row 3 Row 4 …..

Given the Array declaration: ARRAY [lower1 .. upper1, lower2 .. upper2] INTEGER

The address of Array element ARRAY [s1, s2] is calculated:

```
W * [(s1 - lower1) * (upper2 - lower2 + 1) + (s2 - lower2)]
```

where W is INTEGER Word size in bytes, (MIPS = 4).

Given the array address calculation method above, and this Array declaration:

```
X,Y : ARRAY [1..25,1..15] INTEGER
```

Generate the quadruples for this program code fragment:

Note: K is a declared integer variable

```
FOR N :=  1 TO 25 DO

    X[N,2*K+1]  := Y[N, 2*K]
```

Note : The quadruples (about 20) are created as the code fragment is parsed, and the addressing expression is expanded.

[50 points]

2018

sc

FOR Loop

```
:=      #1            N
JGT     N     #100   ( )
     - - - Body - - -
+       #1     N      i1
:=      i1            N
JMP                  ( )
```

Address Calc

$$W * \left[(s1 - lower1) * (upper2 - lower2 + 1) + (s2 - lower2) \right]$$

```
X,Y : ARRAY [1..25, 1..15] INTEGER

X[N, 2K+1]  :=  Y[N, 2K]
```

Next Page ⇒

sc

$$X[N, \underbrace{\underbrace{2 * K}_{i2} + 1}_{i3}] := Y[N, \underbrace{2 * K}_{i2}]$$

*	#2	K	i2
+	i2	#1	i3

$X[N, i3]$

$$\underbrace{4 * [\underbrace{\underbrace{\underbrace{(N-1)}_{i6} * (\underbrace{\underbrace{15 - 1}_{i4} + 1}_{i5})}_{i7} + (\underbrace{i3 - 1}_{i2 \ \checkmark})}_{i8}}_{i9}$$

−	#15	#1	i4
+	i4	#1	i5
−	N	#1	i6
*	i6	i5	i7
+	i7	i2	i8
*	#4	i8	i9

$$X[N, 2K+1] = X[i9]$$

NEXT PAGE ⇒

SC

Y[N, 2*K]

$$\underbrace{4 * [\underbrace{\underbrace{\underbrace{(N-1)}_{i6\checkmark} * \underbrace{(15-1+1)}_{i5\checkmark}}_{i7\checkmark} + \underbrace{(i2-1)}_{i10}}_{i11}]}_{i12}$$

−	i2	#1	i10
+	i7	i10	i11
*	#4	i11	i12

Y[N, 2K] = Y[i12]

NEXT PAGE ⇒

SC

Final Quadruples

1	:=	#1		N
2	JGT	I	#100	(18)
3	*	#2	K	i2
4	+	i2	#1	i3
5	−	#15	#1	i4
6	+	i4	#1	i5
7	−	N	#1	i6
8	*	i6	i5	i7
9	+	i7	i2	i8
10	*	#4	i8	i9
11	−	i2	#1	i10
12	+	i7	i10	i11
13	*	#4	i11	i12
14	:=	Y[i12]		X[i9]
15	+	#1	N	i1
16	:=	i1		N
17	JMP			(2)
18				