Problem Set 10

Justin Ely

605.204.81.FA15 Computer Organization

10 November, 2015

Quadrules Workshop

LC	Operation	Op1	Op2	Result
(1)	:=	#3		K
(2)	:=	#1		I
(3)	BGT	I	#6	(21)
(4)	*	K	#2	i1
(5)	+	i1	#1	i2
(6)	-	i2	#1	i3
(7)	-	I	#1	i4
(8)	*	i4	#9	i5
(9)	+	i5	i3	i6
(10)	*	i6	#4	i7
(11)	*	K	#2	i8
(12)	-	i8	#1	i9
(13)	-	I	#1	i10
(14)	*	i10	#9	i11
(15)	+	i11	i9	i12
(16)	*	i12	#4	i13
(17)	:=	X[i7]		Y[i13]
(18)	+	I	#1	i14
(19)	:=	i14		I
(20)	JMP			(3)
(21)				

Optimization Workshop

The calculation (3L - 1) was done twice for each iteration of the loop. Since the calculation doesn't vary, this was moved outside the loop. Additionally, the 10(I-1) calculation was duplicated in the loop, so this was optimized to be done only once. The new code has many fewer total operations performed.

Other thoughts that came to mind to optimize where looping from 0-6 instead of 1-7. Since I was only used as an index and always required subtraction by 1, this would save 1 operation per loop. Alternatively, I is only ever used as (10I-1), so the loop could begin at 0 and increase by 10 every iteration. This would save two operations per loop. However, these types of changes weren't mentioned in the notes, so I was not sure if this was a type of optimization done. It seems, at first glance, like a rather risky strategy that may be very hard for compilers to accomplish.

LC	Operation	Op1	Op2	Result
(1)	:=	#1		I
(2)	*	#3	L	i1
(3)	-	i1	#1	i2
(4)	BGT	I	#7	(16)
(5)	-	I	#1	i3
(6)	*	i3	#10	i4
(7)	-	i2	#1	i5
(8)	+	i4	i5	i6
(9)	*	i6	#4	i7
(10)	+	i2	i4	i8
(11)	*	i8	#4	i9
(12)	:=	Y[i9]		Y[i7]
(13)	+	#1	I	i10
(14)	:=	i10		I
(15)	JMP			(4)
(16)				