

CDA3101 – F13 – Quiz #3

Mon 14 Oct 2013

Given: `int B[40], i, x ;`
`for i = 0 to 39 do:`
`{ B[i] = 2 * i + 19; x = 3 * B[i] ;`
`if B[i] > 279 then STOP }`

`$s0 ← &B[0]`
`$s1 ← i`
`$s2 ← x`

Q1 (15 pts): Write the above code in MIPS, and fully comment your code.

Q2 (5 pts): Initially, assume the following CPIs:

add, sub = 4 cycles mul = 7 cycles branch = 3 cycles
lw, sw = 3 cycles slt = 4 cycles all others = 1 cycle

Now, if CPI for mul = 3, how many cycles total?

20 pts total – You have 20 minutes to complete

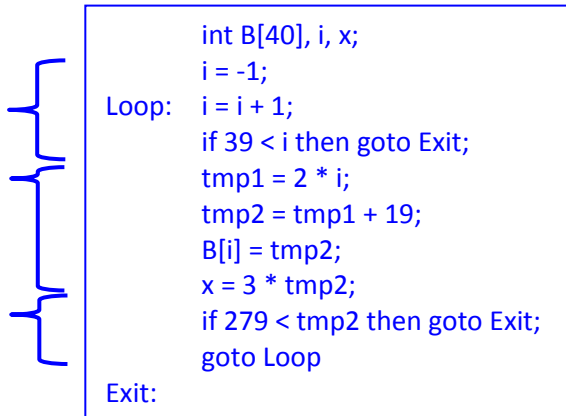
Q1: Let's begin by expanding the high-level language statements into expressions with one operator and an assignment:

`int B[40], i, x ;`

`for i = 0 to 39 do:`

`{ B[i] = 2 * i + 19; x = 3 * B[i] ;`

`if B[i] > 279 then STOP }`



Now we can translate the expanded HLL expressions into MIPS and determine cost ([c]ycles):

	<code>addi \$s1, \$zero, -1</code>	<code># i = -1;</code>	<code>cost = 4c x 1 = 4</code>
	<code>addi \$t3, \$zero, 3</code>	<code># register t3 gets constant 3</code>	<code>4c x 1 = 4</code>
Loop:	<code>addi \$s1, \$s1, 1</code>	<code># i = i + 1;</code>	<code>4c x 41 = 164</code>
	<code>slti \$t0, \$s1, 39</code>	<code># if 39 < i then goto Exit</code>	<code>4c x 41 = 164</code>
	<code>bne \$t0, \$zero, Work</code>		<code>3c x 41 = 123</code>
	<code>j Exit</code>	<code># STOP only happens once</code>	<code>1c x 1 = 1</code>
Work:	<code>sll \$t1, \$s1, 1</code>	<code># tmp1 = 2 * i;</code>	<code>1c x 40 = 40</code>
	<code>addi \$t2, \$t1, 19</code>	<code># tmp2 = tmp1 + 19;</code>	<code>4c x 40 = 160</code>
	<code>sll \$t4, \$s1, 4</code>	<code># offset [in \$t5] = i [in \$s1] x 4</code>	<code>1c x 40 = 40</code>
	<code>add \$t4, \$t4, \$s0</code>	<code># add base addr to offset</code>	<code>4c x 40 = 160</code>
	<code>sw \$t2, 0(\$t4)</code>	<code># B[i] ← tmp2;</code>	<code>3c x 40 = 120</code>
	<code>mult \$t2, \$t3</code>	<code># 3 * tmp2; [constant 3 from t3, above]</code>	<code>7c x 40 = 280</code>
	<code>mflo \$s2</code>	<code># x ← 3 * tmp2;</code>	<code>1c x 40 = 40</code>
	<code>slti \$t5, \$t2, 279</code>	<code># if 279 < tmp2 then goto Exit;</code>	<code>4c x 40 = 160</code>
	<code>beq \$t5, \$zero, Loop</code>	<code># goto Loop</code>	<code>3c x 40 = 120</code>
Exit:			
		TOTAL CYCLES	1,580

Observe that the last two statements are inelegant, but they work because we know the loop will iterate to completion (40 times) since $3(39) + 19 < 279$. (A more careful coding would use the negative logic in the loop limit test earlier in the program.)

Q2: Since there are 40 multiplications in the MIPS realization of Q1, if we incur 3 cycles per mult instead of 7, then we save $4 \text{ cycles} \times 40 \text{ iterations} = 160 \text{ cycles}$. So the total number of cycles becomes $1,580 - 160 = 1,420$, for a total savings of $160/1580 = 10.1 \text{ percent}$.

Note: We could have coded the expression $x = 3 * tmp2$ in terms of two additions (**add \$t6, \$t2, \$t2 ; add \$t6, \$t6, \$t2**), but that would be bad programming practice, for two reasons:

1. The cost of two additions would be 8 cycles (per the givens), versus 7 cycles for a multiplication, so we would be designing a penalty into the program; and
2. The two additions would not benefit from the cost reduction for the multiplication, so we are blocking any further optimization of the program.

Finally, observe that the statement **tmp1 = 2 * i** is coded with an **sll** (shift left logical) in MIPS, instead of a **mult**. This is good practice, because the **sll** consumes one cycle (from the givens), in contrast with the **mult** that consumes 7 cycles (initially, then 3 cycles after optimization).

