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 }

```
        int B[40], i, x;
        i = -1;
Loop:   i = i + 1;
        if 39 < i then goto Exit;
        tmp1 = 2 * i;
        tmp2 = tmp1 + 19;
        B[i] = tmp2;
        x = 3 * tmp2;
        if 279 < tmp2 then goto Exit;
        goto Loop
Exit:
```

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

```
        addi   $s1, $zero, -1      #  i = -1;                          cost =  4c x  1 =    4
        addi   $t3, $zero, 3       #  register t3 gets constant 3              4c x  1 =    4
Loop:   addi   $s1, $s1, 1         #  i = i + 1;                               4c x 41 =  164
        slti   $t0, $s1, 39        #  if 39 < i then goto Exit                 4c x 41 =  164
        bne    $t0, $zero, Work                                                3c x 41 =  123
        j      Exit                #  STOP only happens once                   1c x  1 =    1
Work:   sll    $t1, $s1, 1         #  tmp1 = 2 * I;                            1c x 40 =   40
        addi   $t2, $t1, 19        #  tmp2 = tmp1 + 19;                        4c x 40 =  160
        sll    $t4, $s1, 4         #  offset [in $t5] = i [in $s1] x 4         1c x 40 =   40
        add    $t4, $t4, $s0       #  add base addr to offset                  4c x 40 =  160
        sw     $t2, 0($t4)         #  B[i] ← tmp2;                             3c x 40 =  120
        mult   $t2, $t3            # 3 * tmp2;  [constant 3 from t3, above]    7c x 40 =  280
        mflo   $s2                 # x ← 3 * tmp2;                             1c x 40 =   40
        slti   $t5, $t2, 279       #  if 279 < tmp2 then goto Exit;            4c x 40 =  160
        beq    $t5, $zero, Loop    #  goto Loop                                3c x 40 =  120
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 cycles x 40 iterations = 160 cycles. So the total number of cycles becomes 1,580 – 160 = 1,420, for a total savings of 160/1580 = 10.1 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).