Assignment 5

CDA 5155 — Dr. Whalley — Spring 2018

David Miller

April 18, 2018

Consider the following code:

```
for (i = 0; i < 500; i++)
   e[i] = (a[i] + b[i]) * (c[i] - d[i])
```

Assume that the processor has a maximum vector length of 64 and the startup overheads of the load/store unit is 15 cycles, the multiply unit is 8 cycles, and the add/subtract unit is 5 cycles.

a. First, strip mine the C source code above so that each inner loop iterates for at most 64 times.

The code given in the course lectures could be easily modified to fit to this problem via `n` ← `500`, `MVL` ← `64`, and `Y[i] = a * X[i] + Y[i]` ← `e[i] = (a[i] + b[i]) * (c[i] - d[i])`:

```
low = 0;
VL = n % MVL;
for (j = 0; j <= n/MVL; j++) {
   for (i = low; i < low+VL; i++)
      Y[i] = a * X[i] + Y[i];
   low += VL;
   VL = MVL;
}
```

⇒

```
low = 0;
VL = 52;
for (j = 0; j <= 7; j++) {
   for (i = low; i < low+VL; i++)
      e[i] = (a[i] + b[i]) * (c[i] - d[i]);
   low += VL;
   VL = 64;
}
```

b and c. Convert the strip-mined C loop into VMIPS assembly code. Assume that Ra, Rb, Rc, Rd, and Re contain the starting addresses of the arrays a, b, c, d, and e, respectively. Further assume all vector and integer VMIPS registers are available for use in the loop. Assuming chaining and a single vector memory unit, how many chimes are required for each iteration of the loop containing the vector operations?

```
             LI        Rn,500         // Load 500 into Rn
             ANDI      R6,Rn,#63      // Get value n % MVL
             MTC1      VLR,R6         // Set first value of VLR to n % MVL
             DSLL      R6,R6,#3       // First offset for addresses
             DSLL      R8,Rn,#3       // Get offset for end of addresses
             DADD      R8,R8,Ra       // Last address for a vector
             MOV       R7,#64         // Move 64 into R7
          Loop:
convoy 1     LV        V1,Ra          // Load a into V1
convoy 2   { LV        V2,Rb          // Load b into V2
           { ADDVV.D   V3,V2,V1       // Store a + b in V3
convoy 3     LV        V1,Rc          // Load c into V1
           { LV        V2,Rd          // Load d into V2
convoy 4   { SUBVV.D   V4,V2,V1       // Store c - d in V4
           { MULVS.D   V1,V3,V4       // Store (a + b) * (c - d) in V1
convoy 5     SV        V1,Re          // Store the result to e
             DADD      Ra,Ra,R6       // Apply address offset to a
             DADD      Rb,Rb,R6       // Apply address offset to b
             DADD      Rc,Rc,R6       // Apply address offset to c
             DADD      Rd,Rd,R6       // Apply address offset to d
             MTC1      VLR,#64        // Set VLR size to 64
             MOV       R6,#512        // Address offset now size of full VLR
             BNE       R8,Ra,Loop     // Keep looping until all elements are exhausted
```

Since we have five convoys we also have five chimes.

d. Again assuming chaining and a single vector memory unit, how many clock cycles are required for each e[i] result on average, including startup overhead? You can assume the scalar MIPS instructions can be overlapped with the vector operations.

We have the following convoy execution times

$$\begin{aligned}
\text{Convoy 1} &: 64 + 15 = 79\\
\text{Convoy 2} &: 64 + 15 + 5 = 84\\
\text{Convoy 3} &: 64 + 15 = 79\\
\text{Convoy 4} &: 64 + 15 + 5 + 8 = 92\\
\text{Convoy 5} &: 64 + 15 = 79\\
\text{Total Convoy Time} &: 79 + 84 + 79 + 92 + 79 = 413
\end{aligned}$$

Now to determine how many loop executions will occur is simply $Num\_Loops = \lceil \frac{\texttt{n}}{\text{MVL}} \rceil = \lceil \frac{500}{64} \rceil = 8$. Finally we get that

$$\overline{exec\_time_{e[i]}} = \frac{Total\_Convoy\_Time}{|e|} \times Num\_Loops = \frac{413}{500} \times 8 = 6.608$$