Chapter 3. Data

Problem 3.1

From Tables 3.2, 3.3, and 3.4:

LS: total bytes $= 200 * 4B = 800B$
R/M scientific: total bytes $= 180 * 3.2B = 576B$, $576/800 = .72$
R/M commercial: total bytes $= 180 * 3.8B = 684B$, $684/800 = .86$
R+M scientific: total bytes $= 120 * 4B = 480B$, $480/800 = .60$
R+M commercial: total bytes $= 120 * 3.8B = 456B$, $456/800 = .57$

Problem 3.2

From Table 3.3: R/M $\rightarrow$ 180 total instructions. From Table 3.15: 1 of these is LDM, 1 of these is STM. Data from section 3.3.1: LDM $\rightarrow$ 3 registers $\rightarrow$ 2 cycles, STM $\rightarrow$ 4 registers $\rightarrow$ 3 cycles. Base CPI = 1.

CPI with LDM + STM $= ((178 * 1) + (1 * 2) + (1 * 3))/180 = 1.0167$.

$1.0167/1 \rightarrow 1.7\%$ slowdown.

Problem 3.5

Using data from Tables 3.4 and 3.14:

Frequencies:

FP total = 12%
Add/Sub = 54%
Mul = 43%
Div = 3%

Assuming that the extra cycles cannot be overlapped, then the penalties can be calculated as:

Add/Sub $= 0.54 \times 1$
Mul $= 0.43 \times 5$
Div $= 0.03 \times 11$

Total excess CPI $= (0.54 + 2.15 + 0.33) \times .12 = 0.362$

Problem 3.6

(See problem 3.5 solution for relative frequencies.) Extra floating point cycles can be overlapped. Penalty occurs only due to data dependency on the results. Using data from Table 3.19:

a. Add/Sub $= 0.403 \times 0.54 \times 1 = 0.218$ cycles per fpop

b. Mul $= (0.403 \times 5 + 0.147 \times 4 + 0.036 \times 3 + 0.049 \times 2 + 0.067 \times 1) \times 0.43 = 1.24$ cycles per fpop

c. Div $= (0.404 \times 11 + 0.147 \times 10 + 0.036 \times 9 + 0.049 \times 8 + 0.067 \times 7 + 0.017 \times 6 + 0.032 \times 5 + 0.025 \times 4 + 0.028 \times 3 + 0.022 \times 2 + 0.035 \times 1) \times .03 = 0.229$ cycles per fpop

Total excess CPI $= (0.218 + 1.24 + 0.229) \times .12 = 0.202$

Problem 3.10

From Table 3.4, fixed point operations account for 16 instructions per 100 HLL operations. From Table 3.14, fixed point multiply accounts for 15% of all fixed point instructions. Number of fixed point multiply operations per 100 HLL is $.15 \times 16 = 2.4$.

When implementing multiplication by shifts and adds, each fixed point multiply requires on average 13 shifts, 5 adds, and 22 branches. Total operations/multiply = 40.

Extra shifts/100 HLL = $13 \times 2.4 = 31.2$

Extra adds/100 HLL = $5 \times 2.4 = 12$

Extra branches/100 HLL = $22 \times 2.4 = 52.8$

Table 1: Instructions per 100 HLL ops

Move		107
Branch	$= 26 + 52.8 =$	78.8
Floating point	$= 24 - 2.4 =$	21.6
Fixed point	$= 16 + 12 =$	28
Shifts	$= 27 + 31.2 =$	58.2
Total		293.6

Expected instructions per 100 HLL operations = 293.6.

Problem 3.13

a. 256/256

$19\% + 42\% = 61\%$.

b. 384/128

For 384: capture rate $= (384 - 256) * ((22 - 19)/(512 - 256)) + 19 = 20.5\%$.

$20.5\% + 39\% = 59.5\%$.

c. 448/64

For 448: capture rate $= (448 - 256) * ((22 - 19)/(512 - 256)) + 19 = 21.25\%$.

$21.25\% + 35\% = 56.25\%c)448/64$.

d. 480/32

For 480: capture rate $= (480 - 256) * ((22 - 19)/(512 - 256)) + 19 = 21.625\%$.

$21.625\% + 32\% = 53.62\%$.

e. 496/16

For 496: capture rate $= (496 - 256) * ((22 - 19)/(512 - 256)) + 19 = 21.8125\%$.

$21.81\% + 32\% = 53.81\%$.

Problem 3.14

From Table 3.10, 80% of branches are conditional. From Table 3.4, 26 branch instructions per 100 HLL instructions. From Table 3.3, 180 total instructions per 100 HLL instructions. For conditional branch: if mean interval > 1, branch takes 1 cycle. If mean interval < 1, branch takes 2 cycles.

Assume all branches take 1 cycle: CPI $= 1$.

Assume all conditional branches take 2 cycles:

CPI $= ((26 * 0.8) * 2 + (180 - (26 * 0.8) * 1))/180 = 1.12$.

Problem 3.15

```
 1) ADD.W R7, R7, 4
 2) LD.W R1, 0(R7)
3) MUL.W R2, R1, R1
4) ADD.W R3, R3, R2
 5) LD.W R4, 2(R7)
6) SUB.W R5, R2, R3
7) ADD.W R5, R5, R4
```

Address interlocks have a 4 cycle delay, and execution interlocks have a 2 cycle delay, all when interlocked by the preceding instruction.

I2 has an address interlock from I1 through R7. This requires 4 extra cycles.

I3 has an execution interlock on I2 through R1. This requires 2 extra cycles.

I4 has an execution interlock on I3 through R2. This requires 2 extra cycles.

I5 requires R7 from I1, but due to a previous interlock, this result is already available.

I6 has an execution interlock on I4 through R3. Because I6 is 2 instructions away, this requires 1 extra cycle. I6 does not interlock on R2 because of a previous interlock.

I7 has an execution interlock on I6 through R5. This requires 2 extra cycles.

Total cycles (issue) $= 7$ cycles

Total extra cycles $= 4 + 2 + 2 + 1 + 2 = 11$ cycles

Total time to execute $= 18$ cycles.