Errata and Clarifications
for
Computer Architecture: Pipelined and Parallel Processor Design
1st and 2nd printings

Please consult the bottom of the page facing the Contents page. The bottom line indicates the printing number of your text. Items marked with * were corrected in the second printing. (The Indian edition is a first printing.)

p. 23* in the heading for Table 1.6, the (R+M) should be (R/M).

p. 161* in Example 3.1 (a), in the line that ends with (Table 3.10c), the '7.25' should be '.725'.

p. 188* the timing template that shows IA–DF–D– should show IA–IF–D–.

p. 721* Table A.4 entries for 4K, 64B line should be 1.4500, 1.1400, 1.0300.

---

p. 79 Study 2.2 concludes that $\Delta t = 14\,\text{ns}$ and $s = 9$ is optimum. Better solutions exist outside the range of Δt evaluated, specifically $\Delta t = 10\,\text{ns}$ and $S = 13$ gives total instruction execution of 130 ns and $G = 29.4$ MIPS.

p. 339 In equation 6.3 and last equation on page, the "$\lceil\ \rceil$" mark was shifted. In equation 6.3, both L/v instances should be $\lceil L/v \rceil$. In the last equation, both terms should have $\lceil \frac{L}{m \cdot v} \rceil$.

p. 440 Note: The discussion on pp. 440–41 on finding a word in a module is misleading. A much simpler method is described below.

In finding the address of a word in a module, note that $2^k \pm 1$ is always relatively prime to 2^n. Now we also know that residues of relatively prime modules are unique up to the product of the moduli (this is the Chinese Remainder theorem). So that an address, represented by the pair (a_1, a_2)

$$
\begin{aligned}
a_1 &= A \bmod (2^k + 1) \qquad \text{and} \\
a_2 &= A \bmod 2^m \qquad (2^m \text{ the size of a memory module})
\end{aligned}
$$

is unique up to $(2^k + 1)2^m$, which is simply the size of memory.

So all that has to be done is to find $A \bmod (2^k + 1)$ and use it as the module address, then use the low order m bits of A as a word address in a module and we have a unique address pair (a_1, a_2) that spans the address space of $2^k + 1$ modules, each containing 2^m words.

Example

Consider a memory consisting of 5 modules of 4 words each (representing 20 words, 0 through 19).

A mod 4 word address	A mod 5 module address 0	1	2	3	4
0	0	16	12	8	4
1	5	1	17	13	9
2	10	6	2	18	14
3	15	11	7	3	19

That is, address 11 will be contained in word 3 (11 mod 4) and module 1 (11 mod 5). Of course, the above pairing works equally well for pairs of the form (a_1, a_2):

$$\begin{aligned} a_1 &= A \bmod 2^k - 1, \\ a_2 &= A \bmod 2^m. \end{aligned}$$

For the problem at hand—managing stride—clearly some mods of the form $2^k \pm 1$ are better than others, as they are prime. Thus, if about 4 memory modules were required, we would choose 5 as the prime number of modules.

Similarly, we have:

Approximate number of modules required (determined by Bw requirements)	Choose prime of form $2^k \pm 1$
4	5
8	7
16	17
32	31
64	—

p. 491 in Example 7.8, following $\delta = n/z$, delete parenthetical remark. The proper explanation is found on top of page 492. On the last line, $\gamma^w_{\text{opt}} = 0.094$, not 0.94.

p. 492 The second line should be $B(4, 2, 0.094, .25)$. The remaining results are correct as presented.

p. 537 The SRMP analysis (bottom of page) refers to Figure 8.15b.

p. 616–617 The discussion of the error between the low population and the asymptotic model could be misleading. This section should include a cautionary note:

> Note that a 10% error in occupancy can cause significantly larger errors in queue size if the occupancy is large (e.g., $\rho > .8$). Care should be used if the asymptotic model gives a result of $\rho_a > .8$. In these cases, one should consider the low population model for cases in which $n \leq 10$ or even 20.

p. 751 t_i is the time between successive departures, not arrivals.