Errata for CS:APP3e and its Instructors Manual

Last updated 07/31/2018.

Despite our best efforts to create a book with zero defects, our vigilant readers have pointed out some bugs. Please report any new errata to Randy Bryant and Dave O'Hallaron. Note that some of these errors have been corrected in more recent printings.


- Preface
- Chapter 1: A Tour of Computer Systems
- Chapter 2: Representing and Manipulating Information
  - p. 45, code for show_bytes. Variable i should be declared to have type size_t. 
    Posted 07/11/2015. Randal E. Bryant
  - p. 47, aside “New to C? Formatted printing with printf,” second paragraph. The referenced data type should be int32_t, not int_32t. 
    Posted 05/23/2016. Yili Gong
  - p. 59, Aside at top of page. The second-to-last sentence should state: “This behavior is not guaranteed for C programs, however, and so shift amounts should be kept less than the number of bits in the value being shifted.” 
    Posted 09/28/2015. Ruth Anderson
  - p. 71, first full paragraph, line 6. The range should be 0 ≤ x ≤ Umax_w. 
    Posted 10/25/2015. Shoeb Mohammed
  - p. 82, second line. ‘x should be x’. 
    Posted 10/25/2015. Shoeb Mohammed
  - p. 82, third line. Value x should be computed with function $B2U_w(x)$, not $B2T_w(x)$. 
    Posted 11/13/2015. Parinya Suparit
  - p. 82, first equation in DERIVATION. The left-hand side should be computed with function $B2U_w(x)$, not $B2T_w(x)$. 
    Posted 11/13/2015. Parinya Suparit
  - p. 93, Figure 2.26. In the axis labels, the minus signs are incorrectly printed as the numeral ‘2’. A correct version of the image can be found here. 
    Posted 09/23/2016. Matt Toups
  - p. 111, Practice Problem 2.45. The entry in the third row should be 25/16. 
    Posted 11/23/2015. Mathieu Borderé
  - p. 125, Line before Practice Problem 2.54. The number shown should be -2147483648 rather than -21483648. 
    Posted 09/05/2016. JiaSheng Chen
  - p. 153, Solution to Problem 2.32. The sentence starting on third line should state “In fact, the opposite is true: $tsub-ok(x, TMin)$ should yield 1 when x is negative and 0 when it is nonnegative.” 
    Posted 10/25/2015. Shoeb Mohammed
  - p. 154, Solution to Problem 2.35, second line of part 3. It would be more precise to state $|r| < |x| \leq 2^{w-1}$. (Note that the argument only requires $|r| < 2^w$, and so the original statement and proof are valid.) 
    Posted 07/12/2018. Wang Lei
- Chapter 3: Machine-Level Representation of Programs
  - p. 171, second-to-last line. $2^{48}$ is 256 terabytes, not 64 terabytes. 
    Posted 09/09/2016. Don Bagert
  - p. 174, code annotation at top of page. It should read “Disassembly of function multstore in binary file mstore.o.” 
    Posted 10/25/2015. Shoeb Mohammed
  - p. 175, code annotation in middle of page. It should read “Disassembly of function multstore in binary file prog.” 
    Posted 10/25/2015. Shoeb Mohammed
• p. 177, aside “ATT versus Intel assembly-code formats,” last sentence of first paragraph. The assembly code shown is for the function `multstore`, not `sum.`
  
  *Posted 11/09/2015. Shoeb Mohammed*

• p. 179, Section 3.4, first paragraph. The ranges given for the eight named registers for the three different instruction sets are incorrect. They should be:
  
  - For 8086: `%ax` through `%sp`.
  - For IA32: `%eax` through `%esp`.
  - For x86-64: `%rax` through `%rsp`.
  
  *Posted 07/19/2017. Yili Gong*

• p. 183, sample code near bottom of page, line 4. The code should be “`movb $-17, (%rsp)`.”
  
  *Posted 10/24/2015. Jason Waterman*

• p. 184, (Clarification, not an erratum) Figure 3.5. Although there is an instruction `movzbg`, the GCC compiler typically generates the instruction `movzb` for this purpose, relying on the property that an instruction generating a 4-byte with a register as destination will fill the upper 4 bytes of the register with zeros.
  
  *Posted 04/27/2018. Randal Bryant*

• p. 188, aside “Some examples of pointers,” call to `printf`. The format string should be `"a = %ld, b = %ld\n".`
  
  *Posted 11/09/2015. Shoeb Mohammed*

• p. 191, second full paragraph. The first sentence should state: “The third column of Figure 3.9 illustrates the effect of executing the instruction `popq %rdx` …”
  
  *Posted 09/28/2015. Max Ma*

• p. 191, fourth line of paragraph that begins “The third column of Figure 3 …” It should state that the value `0x123` remains at memory location `0x100`.
  
  *Posted 11/09/2015. Carlos Galdino*

• p. 198, first line of first full paragraph. There should be a period after the word “forms”.
  
  *Posted 09/19/2016. Anise Ghorbani*

• p. 199, third full paragraph. The reference to instruction `idivl` should be to `idivq` instead.
  
  *Posted 02/21/2017. Changan Wang*

• p. 199, fourth full paragraph. The paragraph should start with “For most applications of 64-bit division …”
  
  *Posted 09/19/2015. Xingda Zhai*

• p. 200, first line of text. It should state “In this code, argument `qp` must first be saved in a different register (line 2), …”
  
  *Posted 08/16/2015. Dmitry Neverov*

• p. 201, 9 lines from bottom. For the shift instruction, it should state: “The overflow flag is changed only when the shift amounts is 1, following rules that depend on the shift type.”
  
  *Posted 10/30/2017. Jiaheng Wang*

• p. 204, Practice Problem 3.13, second paragraph. It should state “Suppose `a` is in some portion of `%rdi` while …”
  
  *Posted 10/15/2015. Ronald Greenberg*

• p. 205, Practice Problem 3.13, code for D. The second instruction should be “`setne %al`.”
  
  *Posted 11/09/2015. Carlos Galdino*

• p. 214, third paragraph. The second sentence should be “The function computes the absolute value of the difference of its two arguments `x` and `y`, …”
  
  *Posted 11/11/2015. Yili Gong*

• p. 214, Figure 3.17(c). The comment on line 8 of the code should state “Return `rval`”.
  
  *Posted 03/18/2017.*

• p. 218, assembly code for `cread`. In the annotations for lines 3 and 5, “x” should be “xp”.
  
  *Posted 11/19/2015. Vlad Buslov*

• p. 226, last sentence before Practice Problem 3.25. It should state “The compiler has determined that the loop can only be entered when `n` > 1, and that, as `n` is decremented, it will reach value 1 before reaching a value less than 1.”
  
  *Posted 06/11/2017. Daniel O’Brien*

• p. 234, first and second lines of text. They should state “ … indexed by register `%rsi`, which holds …”
  
  *Posted 09/02/2015. Dmitry Neverov*

• p. 235, Figure 3.23, annotation for line 5 of code. It should state “Goto *jt[index].”
  
  *Posted 12/25/2015. Xinzhen Chen*
• p. 238, Problem 3.31, Comments in assembly code at top of page. The register allocation is listed incorrectly. It should state “a in %rdi, b in %rsi, c in %rdx, dest in %rcx.”
  Posted 07/25/2015. Yu Zhong and Carlos Galdino

• p. 241, fourth and fifth lines. They should state “Procedure P can pass up to six integral values (i.e., pointers and integers) in registers...”
  Posted 09/02/2015. Dmitry Neverov

• p. 243, Figure 3.27a, line 2. The annotation should state “L1: y+2.”
  Posted 02/15/2016. Xinzheng Chen

• p. 244, first full paragraph. The value 0x400054e should be 0x40054e.
  Posted 07/26/2017. Daniel O’Brien

• p. 244, second full paragraph. The value 0x4000560 should be 0x400560.
  Posted 07/26/2017. Daniel O’Brien

• p. 244, Comment after line 4. It should state “Disassembly of first(long x)”
  Posted 09/02/2015. Dmitry Neverov

• p. 249, Fourth line from bottom. Sentence ending “... and 17 (s3).” should be “... and 17 (x4).”
  Posted 11/29/2016. Hyun Chun

• p. 252, Practice Problem 3.34. The first two sentences should state “Consider a function P, which generates local values, named a0–a7. It then calls function Q, which has no arguments.”
  Posted 11/23/2015. Vlad Buslov

• p. 257, Table near the bottom of the page. The assembly code for the first entry should be “movq %rdx, %rax.”
  Posted 10/03/2015. Zhiwei Xin

• p. 259, Figure 3.36 The fourth block from the bottom should have the label A[3][2].
  Posted 04/04/2017. Dean Kajmakci

• p. 259, assembly code in middle of page, line 3. The annotation should state: "Read from M[xA+12i + 4j]."
  Posted 10/25/2015. Karan Dwivedi

• p. 263, assembly code in middle of page, line 3. There is a missing ‘)’ at the end of the annotation.
  Posted 02/11/2016. Elizabeth White

• p. 266, middle of page. The declaration and initialization of r should be:
  
  struct rect r = { 0, 0, 10, 20, 0xFF00FF };
  
  Posted 08/04/2015. Yu Zhong

• p. 277, first bullet, second and third lines after declarations. They should state "... if the object has type T, then the pointer has type T *.”
  Posted 09/02/2015. Dmitry Neverov

• p. 281, Figure 3.40. The first sentence of the caption should read “Character array buf is just below part of the saved state.”
  Posted 12/01/2015. Yili Gong

• p. 282, Table with heading “Characters typed.” The second entry should have the range 8-23.
  Posted 11/6/2017. Weicheng Pei

• p. 282, Practice Problem 3.46, second paragraph. It should state that get_line is called with return address 0x400076.
  Posted 11/23/2015. Vlad Buslov

• p. 284, Practice Problem 3.46, part E. The question for ask for three things wrong with the code.
  Posted 07/26/2017. Daniel O’Brien

• p. 293, Practice Problem 3.49, second paragraph. The text should state: “... there may be an offset of e2 bytes between the values of s2 and p.”
  Posted 11/09/2015. Shoeb Mohammed

• p. 295, Figure 3.45. The lower 128 bits of register %ymm13 should be labeled %xmm13.
  Posted 10/28/2015. David Hirschv

• p. 300, Problem 3.51, second line. The return value has type "dest_t.”
  Posted 10/13/2016. Anise Ghorbani

• p. 306, Table listing two instructions. These should be vucomiss and vucomisd (AVX instructions), rather than ucomiss and ucomisd (SSE instructions).
  Posted 03/09/2018. Weicheng Pei

• p. 313, Problem 3.60, first line of code. Argument n should be declared as being of type int.
  Posted 07/21/2015. Lauren Cooper
• p. 326, Solution to Problem 3.3, third line of text. These examples do not rely on the rules of integral promotion. These rules concern operations performed on data values smaller than those of type int. See, for example https://wiki.sei.cmu.edu/confluence/display/c/INT02-C.+Understand+integer+conversion+rules. This topic is not covered in the book.  
  
  Posted 04/26/2018. Randal Bryant

• p. 326, Solution to Problem 3.3, fifth line of code. The code should be movq %rax, $0x123 to be consistent with the problem statement, although both versions have the same error.  
  
  Posted 06/02/2015. Zhi Li

• p. 326, Solution to Problem 3.3, sixth line of code. The code should be movl %eax, %rdx to be consistent with the problem statement, although both versions have the same error.  
  
  Posted 09/02/2015. Elizabeth White

• (Clarification, not an erratum) p. 326, Solution to Problem 3.4, seventh line of code. The GCC compiler generates the instruction movzb1 for this case, even though the goal is to extend the 1-byte value to 8 bytes. See the note on Figure 3.5 (p. 184).  
  
  Posted 04/27/2018. Randal Bryant

• p. 330, Solution to Problem 3.14D. The comparison is for <=.  
  
  Posted 08/16/2015. Dmitry Neverov

• p. 331, Solutions to Problems 3.15B and 3.15D. The values starting with "0x0x" should start with "0x" instead.  
  
  Posted 12/04/2015. Carlos Galdino and Yili Gong

• p. 333, Solution to Problem 3.20. The comment on the third line of code should read "Test x".  
  
  Posted 06/02/2015. Curtis Gagliardi

• p. 338, Solution to Problem 3.30, third bullet. The final sentence should say "Thus, case labels 3 and 6 are missing in the switch statement body.”  
  
  Posted 09/27/2015. Xinyun Zhao

• p. 339, Solution to Problem 3.32, line labeled “F4”. The instruction should be “repl retq.”  
  
  Posted 10/13/2016. Wenjun Huang

• p. 346, Solution to Problem 3.46C. The program is attempting to return to address 0x400034, not 0x400034.  
  
  Posted 10/24/2016. Wenjun Huang

• p. 347, Solution to Problem 3.47E. In addition, the program should test the value returned by gets to make sure it’s not NULL.  
  
  Posted 07/26/2017. Daniel O’Brien

• p. 348, Solution to Problem 3.51. In the third entry of the table, the conversion is from double to float, as is requested in the problem statement.  
  
  Posted 08/09/2015. Yu Zhong

• p. 349, Solution to Problem 3.53, last paragraph of text. The ambiguity arises from the commutativity of addition, not multiplication.  
  
  Posted 11/09/2015. Shoeb Mohammed

• Chapter 4: Processor Architecture

  • p. 360, Problem 4.2A. There should be an additional byte with hexadecimal value 00 at the end of the sequence.  
  
  Posted 08/16/2016. James Timmins

  • p. 360, Problem 4.2B. There should be an additional byte with hexadecimal value 90 at the end of the sequence.  
  
  Posted 12/01/2015. Shoeb Mohammed

  • p. 365, Figure 4.6, Line 9 of x86-64 code. The comment should say "Test count".  
  
  Posted 01/02/2017. Lee Wintervold

  • p. 366, middle of page. The four words listed are incorrect. They should be 0x000d000d000d, 0x00c000c000c0, 0x0b000b000b00, and 0xa000a000a000  
  
  Posted 06/23/2017. Yiling Gong

  • p. 371, Practice Problem 4.7. The annotation for line 7 of the code should state “Return 0 or 8.”  
  
  Posted 12/02/2015. Alex Knaust

  • p. 377, Figure 4.13(b): The last selection in the HCL code should read "1 : B;" (The number 1 rather than the letter 1).  
  
  Posted 06/02/2015. Paul Anagnostopoulos

  • p. 404, Figure 4.26, 7th entry. The name should be IOPQ, not IOPL.  
  
  Posted 10/24/2016. Wenjun Huang

  • p. 404, Figure 4.26, 14th entry. The name should be RRSP, not RESP.  
  
  Posted 08/26/2015. Yu Zhong
• p. 405, Caption of Figure 4.27. It should state “Ten bytes are read from the instruction memory ...”
  Posted 07/25/2015. Randal E. Bryant

• p. 407, parenthetical remark before HCL code for srcA. Should state “recall that RRSP is the register ID of %rsp.”
  Posted 08/26/2015. Yu Zhong

• p. 420, second line from bottom. Reference to Figure 4.38(c) should be to Figure 4.38(d).
  Posted 07/23/2018. Jong-won Choi

• p. 439, fourth and fifth lines from bottom. The sentence should state “The value for register %rbx is also forwarded from the memory to the decode stage.”
  Posted 08/09/2015. Liemin Ma

• p. 434, Figure 4.47. The instruction at address 0x016 should be addq.
  Posted 11/23/2015. Warren Crasta

• p. 441, Figure 4.53. The instruction at address 0x32 should be addq %rbx,%rax.
  Posted 07/28/2015. Randal Bryant

• p. 443, Figure 4.55. Three errors:
  • The program is prog6.
  • The instruction at address 0x000 should be irmovq Stack,%rsp.
  • The instruction at address 0x013 should be irmovq $10,%rdx.
  Posted 07/28/2015. Randal Bryant + 12/21/2017 Dave Ohlsson

• p. 444, Figure 4.56. The instructions at addresses 0x016 and 0x020 should be irmovq.
  Posted 10/24/2016. Wenjun Huang

• p. 445, first complete paragraph. As a general note, our processor designs do not attempt to set the program counter (PC) to a consistent value when an exception occurs. In a more complete implementation, the PC would be set to the address of the instruction that causes the exception, but this is not the case for the pipelined implementations.
  Posted 06/05/2018. Eadren King

• p. 454, last sentence on page. It should read “In this figure, you can also see that many of the values in pipeline registers M and W ...”
  Posted 12/25/2015. Jiwen He

• p. 473, Bibliographic Notes. In the last line, “Intel-compatible x86-64 processor” should be “Intel-compatible IA32 processor.”
  Posted 05/04/2015. Scott Wright

• p. 492, Solution to Problem 4.35. Second to last sentence should state: “... the conditional move source value 0x123 gets forwarded into ALU input valA, while input valB correctly gets operand value 0x321.”
  Posted 07/21/2015. Liemin Ma

• p. 494, Solution to Problem 4.44. The answers to part A and part C are both off by 2 instructions:
  • The inner loop of the code using the conditional jump contains 9 instructions, 8 of which are executed when the array element is positive.
  • The inner loop of the code using the conditional move contains 8 instructions.
  • The conditional jump code requires an average of 9.5 cycles.
  • The conditional move code requires an average of 8.0 cycles.
  Posted 12/15/2015. Nathaniel Green

**Chapter 5: Optimizing Program Performance**

• p. 520, around 1/3 way down the page. The reference to register “%eax” should instead be to “%rax.”
  Posted 12/01/2015. Shoeb Mohammed

• p. 533, second to last line in first paragraph. It should state “... up to a factor of 6.”
  Posted 07/26/2017. Daniel O’Brien

• p. 533, code annotations for inner loop of combine5. Variable limit is stored in register %rbp.
  Posted 12/01/2015. Shoeb Mohammed

• p. 533–534, sentence spanning these two pages. It should state: “The loop unrolling leads to two vmulsd instructions—one to multiply acc by data[i], and the second to multiply acc by data[i+1].”
  Posted 01/02/2016. Mark Morrissey

• p. 546. Web Aside OPT:SIMD. The assembly code should reference register “%rcx,” not “%rcs.”
  Posted 05/08/2016. Sheldon Guo

• p. 547. Table in description of Web Aside OPT:SIMD.
• The headings under the category “Floating point” should read float and double, rather than int and long.
  
  Posted 07/25/2015. Randal E. Bryant

• p. 556, Figure 5.33 caption. dest should be dst
  
  Posted 07/31/2018. Tanvir Alam

• Chapter 6: The Memory Hierarchy

    
    Posted 01/11/2016. Yili Gong

  • p. 612, [Clarification]: The last sentence of the Cache Hits section refers to Figure 6.22.
    
    Posted 07/31/2018. Dave Ohlsson

  • p. 640, Figure 6.40, line 22. “i++” in for loop should be “i += stride”.
    
    Posted 03/01/2016. Shoeb Mohammed

  • p. 640, Figure 6.40, line 35. sizeof(double) should be sizeof(long)
    
    Posted 07/31/2018. Weicheng Pei

  • p. 645, Figure 6.44. It should be mentioned somewher that the C matrix is assumed to be initially all zeros. Also line 6 of version ijk could eliminate an unnecessary load by replacing C[i][j] += sum with C[i][j] = sum. However, since this line is not in the inner loop, it doesn't affect the analysis either way.
    
    Posted 07/31/2018. Debbie Neft

  • p. 651, Problem 6.29A, The diagram should have 12 boxes instead of 13.
    
    Posted 11/23/2015. Tj Gilbrough

  • p. 661, The solution to Problem 6.4:
    
    For parts A and B, the 1 MB file consists of 2,048 512-byte logical blocks, not 2,000 as stated in the solution.
    
    Posted 07/30/2018. Jonatan Schroeder

    To be precise, part B should include the total transfer time for the blocks, which is 2 x \( T_{\text{max rotation}} = 12 \) ms.
    
    Posted 07/30/2018. Li Qiuqiao

  • p. 662, Solution to Part (c) Problem 6.5, the expected lifetime is \( \sim 17,535 \) years, not 140 years.
    
    Posted 03/01/2016. David Hirsch

• Chapter 7: Linking

  • p. 682. paragraph 2. in the output from foobar5, the values of x and y are corrupted, but the precise values are system-dependent.
    
    Posted 12/16/2015. Yili Gong

  • p. 683, [Clarification]: The variable x referred to in the first paragraph is from foo5.c.
    
    Posted 07/31/2018. Dave Ohlsson

  • p. 690, paragraph above Figure 7.9 that reads "Recall from Section 3.6.3" should be "Recall from Section 3.6.4."
    
    Posted 07/30/2018. Yili Gong and xiejianxun.

  • p. 698, Figure 7.15. The address for the user stack (2^{48}-1) is wrong. The correct address is much smaller, and varies from process to process.
    
    Posted 07/09/2015. Dave O'Hallaron

  • p. 698, Figure 7.15. %esp should be %rsp.
    
    Posted 10/21/2015. Liz White

  • p. 715, Problem 7.8, Module 2. static int main=1 should be static int main=1;
    
    Posted 07/30/2018. Michael Hinton and Michael Ross

• Chapter 8: Exceptional Control Flow

  • p. 735, Figure 8.13. Two errors:
    
    The address for the user stack (2^{48}-1) is wrong. The correct address is much smaller, and varies from process to process.
    
    Posted 07/14/2015. Dave O'Hallaron

    %esp should be %rsp.
    
    Posted 07/31/2018. Dave Ohlsson

  • p. 748. First paragraph. Remove the last sentence: "The only correct assumption is that each possible outcome is equally likely" or replace it with "The only correct assumption is that each possible outcome can indeed occur."
    
    Posted 07/31/2018. Dave Ohlsson
• p. 751, definition of `getenv`, "Returns: pointer to name if it exists" should be "Returns: pointer to value associated with name if it exists".
  Posted 7/30/2018. Yili Gong and xiejianxun.

• p. 752, Practice problem 8.6, output of `.myecho`. "Command-ine" should be "Command-line".
  Posted 7/30/2018. Li Du.

• p. 756, Figure 8.25. `parseline()` should check that the capacity of `argv` (= MAXARGS, defined in Figure 8.23) is not exceeded.
  Posted 7/30/2018. Dave Ohlsson.

• p. 766, Figure 9.26. The kernel portion of the address space is identical for each process. There is no part of the kernel virtual memory that is different for each process.
  Posted 05/08/2018. Godmar Bak.

• p. 796, Figure 9.47. On the top line of the process graph, p1: x=1 should be p2: x=1.

• Chapter 9: Virtual Memory

  • p. 829, Figure 9.26. The relative path name should be `../droh/hello.c` rather than `../home/droh/hello.c`.
    Posted 06/26/2016. Mathieu Bordere.

• Chapter 10: System-Level I/O

  • p. 989, first sentence. Delete the entire sentence that begins with "If the main thread calls pthread_exit, it waits for all other threads to terminate..." In fact, Pthread_exit never waits for other threads to terminate.

• Chapter 12: Concurrent Programming

  • p. 997, The text describing Figure 12.17 is correct, but the Figure itself has two typos:
- The U_i instruction should be `addq $1, %rdx`.
- The S_i instruction should be `movq %rdx, cnt(%rip)`.

Posted 02/22/2018. Dr. Joann J Ordille

- p. 1002, first box. In the declaration of `sem_init`, the middle parameter should be `int pshared`.

Posted 07/31/2018. Dave Ohlsson

- p. 1015, Figure 12.31, code line 25. `sem_init` should be `Sem_init`.

Posted 07/31/2018. Dave Ohlsson

- p. 1023, ten lines from the bottom. "the static next variable" should be "the static next_seed variable".

Posted 07/31/2018. Dave Ohlsson

- p. 1029. The definition of the mutex ordering rule is too strong. It should read simply "a program is deadlock free if each thread acquires its mutexes in the same order." In fact, the order that mutexes are released does not matter, since V operations never block.

Posted 07/30/2018. Einar Rasmussen

- **Appendix A: Error Handling**
- **Index**
  - p. 1076. There should be no index entry for RESP. Instead, it should be for RRSP with the same information.

Posted 08/26/2015. Yu Zhong