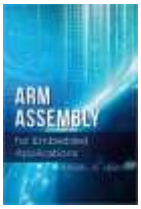


ARM Assembly for Embedded Applications, 1st edition
ISBN 978-1-48357-192-8

The following applies to those who purchased a copy of the textbook online:

Apparently there was a problem in the conversion from MS/Word to the PDF version that was sent to the publisher who then gave it to the online retailers for print on demand sales. The problem is that spaces were somehow inserted to the left of radix points. For example, a number such as 1001.1100 was printed as 1001 .1100. I know that the problem occurs in some of the problems at the end of Chapter 2, but it may very well appear at other places in the text.

CHAPTER 2: BINARY NUMBER SYSTEMS

Page 22, Line 5: Remove the word “is”.

Page 23 (bottom of page) and Page 24 (top of page): Change “Unsigned 8-bit representation of 25_{10}” to “Unsigned 8-bit representation of 20_{10}”.

Page 24, Method 1, Line 1: Remove the second “if”.

Page 24, Method 1, Line 4: Insert “the previous” between “in” and “section”.

Page 24, Method 1, *Warning* near bottom: Remove the word “for”

Page 27, Problem 2: Change "8-bit" to "12-bit".

Page 28, Problem 8: Add the following statement just below the problem:

> *Note: If the process repeats indefinitely, calculate as few digits as necessary to provide same or better resolution.*

Page 28, Problem 10: Change “b_{n-2}” to “$b_{-(n-2)}$”.

Page 29, Problem 15: Add the following statement just below the problem:

> *Note: If the process repeats indefinitely, calculate as few digits as necessary to provide same or better resolution.*

Page 30, Problem 22: Replace the second sentence by the following:

> For input values -128 to -1, display the 2's complement signed representation and label it as "2's complement". For 0 to +127, display the 8-bit representation and label it as "both unsigned and 2's complement". For 128 to 255, display the unsigned representation and label it as "unsigned". Display "out of range" for all other values and terminate the program.

CHAPTER 3: WRITING FUNCTIONS IN ASSEMBLY

Page 32, Figure 3-2: Change "Branch and Exchange" to "Branch Indirect".

Page 34, Figure 3-4: Change the data type of z64 to int64_t.

Page 47, Section 3.7.5: Change section title to “Function GetClockCycleCount”

Page 47, Section 3.8, Line 3: Insert “a” between “are” and “list”.

Page 47: Add the following two sections:

3.7.6 Function PushButtonPressed

Function prototype:

```
int PushButtonPressed(void) ;
```

Returns 1 if the blue pushbutton is pressed, 0 if not.

3.7.7 Function GetRandomNumber

Function prototype:

```
uint32_t GetRandomNumber(void) ;
```

Returns a 32-bit value from the internal random number generator of the Cortex-M4F MCU.

Page 48: Add the following statement just before the first problem:

Note: Use the ADR instruction to load the address of a variable (rather than its contents) into a register, as in ADR R0,a.

CHAPTER 4: COPYING DATA

Section 4.6 (Addressing Modes), page 59, paragraph titled “PC-Relative addressing”:

The use of PC-Relative addressing with the STR instruction and its variants (STRB, STRH, STRD) has been deprecated, similar to the use of PC-Relative addressing with the VSTR instruction as described in Section 9.4.4. Although there are several examples such as `STR R0,x` found in chapter 4, they would actually be rejected by the assembler. To store into a memory location using its label requires a two-instruction sequence such as `ADR R1,x` or LDR R1,=x followed by `STR R0,[R1]`. The ADR instruction is only able to reference locations that are within 4095 bytes of the ADR, which would usually mean that the referenced location resides in flash memory (and therefore cannot be modified during execution). To use a label to reference data in read/write memory thus requires the LDR R1,=x instruction.

Note that since the text uses assembly to write small functions called from a C main program, there is rarely (if ever) a need for PC-Relative addressing because the operands of such functions are typically only the function parameters that are passed to the function in registers and the result is left in a register.

Page 53: Add the following footnote: "The memory operand of LDRD must reside at a mod 4 (word aligned) address to avoid an address alignment fault".

Page 57: Add the following footnote: "The memory operand of STRD must reside at a mod 4 (word aligned) address to avoid an address alignment fault".

Page 68: Add the following footnote: "The memory operands of LDMIA, STMIA, LDMDB and STMDB must reside at a mod 4 (word aligned) address to avoid an address alignment fault".

Page 69, Listing 4-1: Add the following note just below the listing: "The dst and src parameters are assumed to hold word aligned addresses, or else an address fault will occur".

CHAPTER 5: INTEGER ARITHMETIC

Page 83: The minus sign at the end of the second line should not be separated from the digits 24 that appear at the beginning of the next line.

Page 90, second paragraph, second line: Remove the word “limits”.

Page 91, Table 5-5: Remove the entries for non-existent instructions UQADD and UQSUB.

CHAPTER 6: MAKING DECISIONS AND WRITING LOOPS

Page 95, Section 6.1, 2nd paragraph, 4th sentence: The sentence should read, “However, it can also be used to determine if any one of several bits of a register is a 1.”

Page 101, Figure 6-4: Replace the BLE instruction by BNE.

Page 102-103, 107, 109: Comments in code should begin with "//", not with a semicolon.

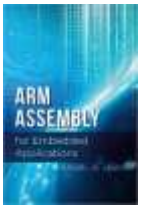
Page 102-103: Replace the last paragraph and subsequent code by the following:

> Since a compare is simply a subtraction that discards the difference but records the characteristics of that result in the flags, one might assume that all that is needed is to simply perform a 64-bit subtraction and check the resulting flags. However, although the N, V and C flags will be correct, the Z flag may not since it will only indicate if the most-significant half of the difference is zero.
>
> When the condition to be tested does *not* depend on the Z flag (GE,LT,HS,LO,MI,PL,VS,VC,AL), then no correction is necessary and the condition test may immediately follow the 64-bit subtraction:

```
...                     // Load operands as before
SUBS      R0,R0,R2      // subtract LS halves, capture borrow
SBCS      R1,R1,R3      // subtract MS halves w/brw; set flags
...                     // OK to test GE,LT,HS,LO,MI,PL,VS,VC,AL
```

> However, all other conditions (EQ,NE,GT,LE,HI,LS) require a different approach. The solution for EQ and NE is quite simple:

```
...                     // Load operands as before
SUBS      R0,R0,R2      // compute LS half of difference
```




```
SBC       R1,R1,R3      // compute MS half of difference
ORRS      R1,R0,R1      // Z=1 iff both halves are zero
...                     // Now OK to test for EQ or NE
```

For LE, GT, LS and HI we can avoid testing the Z flag by reversing the operands. Since x <= y is equivalent to y >= x, this allows us to replace LE by GE or LS by HS, which don't require testing the value of Z:

```
...                     // Load operands as before
SUBS      R2,R2,R0      // subtract LS halves (operands reversed)
SBCS      R3,R3,R1      // subtract MS halves (operands reversed)
...                     // Replace LE/GT by GE/LT, or
...                     // Replace LS/HI by HS/LO.
```

Finally, note that in all cases except EQ and NE, we can avoid modifying one register by replacing the SUBS instruction by a CMP instruction.

Page 103, Section 6.5, 1st paragraph, 4th sentence: Delete the word "condition". The sentence should begin, "The If-Then (IT) instruction …"

Page 104, bullet point 2 in the middle of the page: Change "Only the last instructions…" to "Only the last instruction…" (singular form).

Page 106, Figure 6-6: The "L1:" labels in the last two columns of the second row of cells should be moved up to appear next to the then statements.

Page 109, Section 6.7, 1st paragraph, 3rd sentence: "For and while loop test…" should be "For and while loops test …"

Page 110, Listing 6-2. Change "SUBGT" to "SUBHI" and "SUBLE" to "SUBLS".

Page 110, 1st paragraph: Change "Thumb-2" to "ARM".

Page 111: Replace the code above the last paragraph by:

```
CMP     R0,0    // may be replaced
BEQ     L1      // by CBZ R0,L1

CMP     R0,0    // may be replaced
BNE     L1      // by CBNZ R0,L1
```

Page 112, Problem 1b: Change the initialization in the for loop to "y = 1;".

CHAPTER 7: MANIPULATING BITS

Page 119, Table 7-1, 2nd row, 6th column: Replace "00001101" by "00000101".

Page 119, last paragraph, 3rd line: Change "an" to "a".

Page 127, Listing 7-3:

Replace the code with the following sequence:

```
PackTime:
    LSL     R0,R0,11    // move hour into position at far left
    BFI     R0,R1,5,6   // insert minutes into bits 5-10 of R0
    LSR     R2,R2,1     // divide the seconds by two
    BFI     R0,R2,0,5   // insert seconds/2 into bits 0-4 of R0
    BX      LR
```

Page 128, Listing 7-4:

1st UBFX instruction: Change "25" to "9", Change "25-31" to "15-9"
2nd UBFX instruction: Change "21" to "5", Change "24-21" to "8-5"
3rd UBFX instruction: Change "16" to "0", Change "20-16" to "4-0"

Page 132, 2nd paragraph, 5th line:

Change "0 to 8,388,607 (0 to 8 Mbits)" to "0 to 33,554,431 (0 to 32Mbits-1)"

Page 132, Listing 7-5:

2nd comment line: Change "1 Mbit" to "32 Mbits"
UBFX instruction: Change "data offset" to "byte offset"

Page 133, Problem 1c:

Change the function prototype to: `int64_t ASR64(int64_t s64) ;`

Page 134, Problem 7: Replace the first two sentences in the problem description by "Write a function in ARM assembly language that returns the number of bits in the parameter not including leading and trailing 0's. For example, if the parameter is 006203F016, the function should return the value 19."

Page 135, Problem 9, 1st sentence:

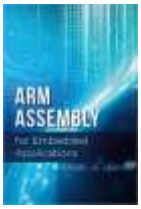
Change "the bit" to "bit 0 of that byte".

CHAPTER 8: MULTIPLICATION AND DIVISION REVISITED

Page 138, Table 8-2, 2nd row: Change "2^6A - 2^1A" to "2^5A – 2^1A".

Page 138, Table 8-2, 2nd row: Change "00111110" to "00011110".

Page 139, Table 8-3, 3rd row: Change "-10" in the Adjusted Dividend column to "-9" and the corresponding entry in the Binary Operand column to "11110111".



CHAPTER 9: GETTING STARTED WITH FLOATING POINT

Page 149, Section 9.1, 2nd paragraph, line 3: Replace "six" by "seven".

Page 153, Section 9.4, 1st paragraph, last line: Delete the word “explain”.

Page 153, Section 9.4, 3rd paragraph, 4th line: Delete the word “in”.

Page 159, Section 9.5, 2nd paragraph, 6th line: Insert the word “to” immediately before “the VCVT mnemonic”.

Page 161, Section 9.6: Change “ARTIHMETIC” to “ARITHMETIC”

Page 162, Table 9-7, last 2 rows: Remove the word "Fused" in both rows, change "VFMA" to "VMLA", and change "VFMS" to "VMLS".

Page 164, Table 9-8, 2nd row: Change "S_d,0" to "S_d,#0.0"

Page 167, Problem 6: Change the parameter "radians" to "x".

CHAPTER 10: WORKING WITH FIXED-POINT REAL NUMBERS

Page 170, 2nd Paragraph, last line: Replace the colon by a period.

Page 178, bottom of the page: Replace the last parenthesized term to ($2^{32}B_{HI} + B_{LO}$)

Page 179, top of the page: Change the last term of the equation from $A_{LO}B_{HI}$ to $A_{LO}B_{LO}$

Page 180, bottom of the page: Inside the box, change the number on the second rule from "1" to "2".

Page 181, Figure 10-4: Change the comment that appears to the right of the B_{63}xA term to "If B<0, subtract A from the most-significant half of unsigned product".

Page 182, middle of the page: Extend the underline left so that it aligns with the $A_{HI}B_{HI}$ term.

Page 188, Programming Problem 3: Change "P R O B L E M 0" to "1".

Page 188, Programming Problem 3: Insert the word "to" between the words "program" and "compute".

Page 188-189, Problems 4, 5, 6: Change “Figure 10-1 and Figure 10-4” to “Listing 10-1 and Listing 10-4”.

Page 189-190, Problems 7, 8, 9: Change the reference to “problem 3” to “problem 6”.

Page 190, Problem 9: Change the parameter "radians" to "x".

CHAPTER 11: INLINE CODE

Page 192, Section 11.2, 3rd paragraph, 3rd line: Replace the phrase "square brackets" with "curly braces".

Page 193, 3rd paragraph that begins, "Note that …": Replace the phrase "square brackets" with "curly braces".

Pages 192 and 193, the asm statements: Replace the square brackets "[…]" with curly braces "{ … }"

Page 194, Second paragraph: Remove the phrase, "preceded by a percent sign". Remove the actual percent sign preceding the left square bracket in the line immediately below the paragraph.

Page 198, code near the bottom of the page: The word "flags" should be moved to the end of the previous line as part of the comment, "// Clobbers R0 and flags".

Page 200, top of the page: Change "`int64_t orig, half ;`" to "`int64_t src, dst ;`".

Page 200, first paragraph, 4th line: Change "… keyword, you …" to "… keyword just before the left parenthesis, you …".

CHAPTER 12: PROGRAMMING PERIPHERAL DEVICES

Page 207, Section 12.1.1, second line: Change "find" to "finds".

Page 211, 1st paragraph, 9th line: Change "$0x40023000_{16}$" to "40023000_{16}".

Page 211, Section 12.1.3, 1st paragraph, 4th line: Change "Listing 12-1" to "Listing 12-2".

APPENDIX B: GRAPHICS LIBRARY FUNCTIONS

Page 233: Change the name of function `ClearDisplay` to `ClearScreen`

Page 233: Add the following two function prototypes just below the prototype for function `SetColor`:

```
void SetForeground(uint32_t color) ; // Same as SetColor

void SetBackground(uint32_t color) ;
```

Page 234: Remove the last parameter ("int alignment") from the function prototype for function DisplayStringAt.

APPENDIX C: TOUCH SCREEN LIBRARY FUNCTIONS

Page 235, Listing C-1: Insert a call to `TS_Init()` immediately before the `while` statement.