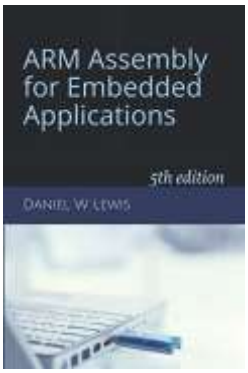



## *Programming Lab 8A*

# Zeller's Rule

Click to download Lab8A-Main.c

*Topics: Replacing division by reciprocal multiplication, replacing multiplication by a sequence of addition, subtraction and shift instructions.*

Prerequisite Reading: Chapters 1-8

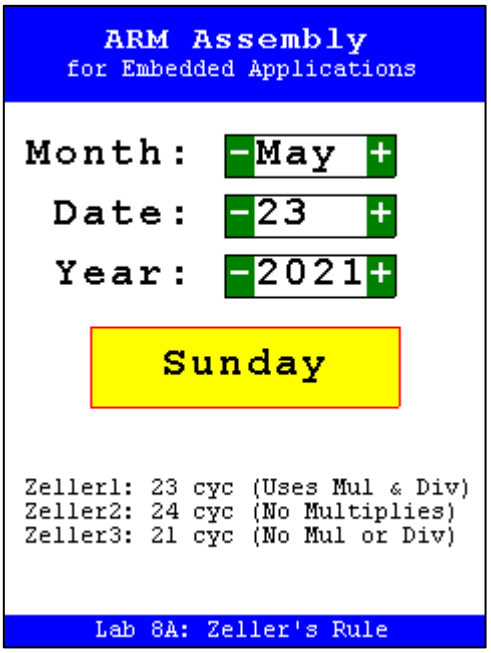
Revised: June 22, 2021

***Background***[1]**:** The following formula is named Zeller's Rule and may be used to convert a date into a day of the week.

$$f = k + \frac{13m - 1}{5} + D + \frac{D}{4} + \frac{C}{4} - 2C$$

*Where:*

- ***k*** is the day of the month.
- ***m*** is the month number. Months must be counted differently for Zeller's Rule: March is 1, April is 2, and so on to February, which is 12. (This makes the formula simpler, because on leap years February 29 is counted as the last day of the year.) Thus, January and February are always counted as the 11th and 12th months of the ***previous*** year.
- ***D*** is the last two digits of the year.
- ***C*** is the century - the first two digits of the year.




***Example:*** January 29, 2064

$$f = 29 + \frac{13 \times 11 - 1}{5} + 63 + \frac{63}{4} + \frac{20}{4} - 2 \times 20$$
$$= 29 + 28 + 63 + 15 + 5 - 40 = 100$$

***Note:*** $D = 63$ even though the year is 2064 because the month is January (see above). All the divisions above are *unsigned* integer divisions that discard any fractional part. We then use *signed* integer division to get the remainder of $f$ divided by 7 since $f$ can be negative. If the remainder is negative, add 7. (This result is what your assembly language functions must return.) A result of 0 means Sunday, 1 means Monday, etc. For our example, $100/7 = 14$, remainder 2, so January 29, 2064 falls on a Tuesday.

***Assignment:*** The main program will compile and run without writing any assembly. However, your task is to create equivalent replacements in assembly language for the following three functions found in the C main program. The original C versions have been defined as "weak" so that the linker will automatically replace them in the executable image by those you create in assembly; you do not need to remove the C versions. This allows you to create and test your assembly language functions one at a time. `Zeller1` serves as a performance baseline using multiply and divide instructions; `Zeller2` and `Zeller3` are used to illustrate performance differences as the result of replacing multiply and divide instructions by equivalent instruction sequences. All functions should use shifting to implement division by 4 and multiplication by 2.

| Function | Multiplication by 13 | Remainder by 7 | Unsigned Division by 5 | Signed Division by 7 |
|---|---|---|---|---|
| Zeller1 | MUST use MUL | MUST use MLS | MUST use UDIV | MUST use SDIV |
| Zeller2 | Must NOT use MUL | Must NOT use MLS | MUST use UDIV | MUST use SDIV |
| Zeller3 | Must NOT use MUL | Must NOT use MLS | Must NOT use UDIV | Must NOT use SDIV |

Use this webpage to find instruction sequences that will divide by a constant without using a divide instruction. Test your code with the main program. If your code is correct, the display should look like the image shown above. Use the touchscreen to adjust the date and see the corresponding day of the week. Errors will be displayed as white text on a red background.

[1] http://mathforum.org/dr.math/faq/faq.calendar.html