

## Unsigned Division Routines

By: M. Ganesh Raaja

Associated Project: Unsigned Division.zip

Associated Part Family: CY8C25xxx, CY8C26xxx

### Summary

The following Application Note describes how to perform unsigned division for any word length.

### Introduction

The M8C instruction set does not contain a `div` command. As long as a C compiler is used, it is not a problem to perform division operations. The coding will be taken care of by the built in 'C' libraries. But performing division in assembly is definitely a problem. To perform division in PSoC, binary division algorithms have to be used.

There are a number of methods to perform division on binary numbers. Radix 2, Radix 4, Radix 16 are some of the command methods. In this Application Note, the Shift, Subtract and Restoration method is used. Of all the methods, this is the easiest to implement, though it may not be the fastest.

### Unsigned Division Algorithm

The Shift, Subtract and Restore method is used to perform a division operation. The method is explained below.

Consider that  $X / Y$  has to be calculated:

- Register X contains the dividend.
- Register Y contains the divisor.
- One more register Z is made an extension of register X such that the length of Z and X is twice that of X.
- At the end of division, X will contain the result with Z containing the remainder.

### Steps Involved

1. Shift the double register (Z and X) to the left.
2. Make a backup of Z.
3. Subtract Y from Z.
4. If the result is negative, set LSB of X to 0. Then restore Z.
5. If the result is positive, set LSB of X to 1.
6. Repeat steps 1 to 5 "n" number of times where "n" is the word length.

After completion of the above steps:

- X contains the quotient.
- Z contains the remainder.

### Program for 8-Bit Division

Following is a program that implements the above algorithm to perform an 8-bit unsigned division:

```
div8:
    mov [remainder+0],00h
    and F,fbh
    mov [lcount],8
d8u_1:
    rlc [dividend+0]
    rlc [remainder+0]

    mov [temp+0],[remainder+0]
    mov a,[remainder+0]
    sub a,[divisor+0]
    mov [remainder+0],a
    jnc d8_2

    mov [remainder+0],[temp+0]
    and [dividend+0],feh
    jmp chkLcount8
d8u_2:
    or [dividend+0],01h
chkLcount8:
    dec [lcount]
    jnz d8u_1
    ret
```

## Program Analysis

In the first, second and third lines of code, the parameters are initialized. The remainder is made 0, the carry is cleared and the loop counter is set to 8, as an 8-bit division is to be performed.

In lines 4 and 5, the dividend and remainder are rotated one bit to the left, i.e., the MSB of the dividend is shifted to the LSB of the remainder.

In line 6, the remainder is stored in a temporary variable.

In lines 7, 8 and 9, the divisor is subtracted from the remainder.

In line 9, a conditional branching is made. If the result of the subtraction is negative (Carry Flag set), the remainder is restored from the temporary variable (line 11) and the LSB of the dividend is cleared (line 12). Then, the program branches to line 15.

If the result of the subtraction is positive (Carry Flag not set), the LSB of the dividend is set to 1 (line 14).

In line 15, the loop counter is decremented. In line 16, 'If' the loop counter has become zero, the program returns, 'Else' it branches to line 4.

## Performing 16-Bit Division

Once a program for 8-bit division has been written, it is very simple to write a program for a 16-bit division. The following program modifications are necessary for a 16-bit division:

1. While initializing, [remainder+1] and [remainder+0] will be initialized to zero.
2. The loop counter will be initialized to 16.
3. RLC will be carried out through [dividend0] [dividend1] [remainder0] [remainder1].
4. Backup of source [remainder+1] [remainder+0] will be made to [temp+1] [temp+0].
5. Instead of an 8-bit subtraction of [remainder0] - [divisor0], a 16-bit subtraction of source [remainder1:remainder0] - [divisor1:divisor0] will be performed.
6. In place of 8-bit restoration of [remainder], a 16-bit restoration is performed from [temp+1] [temp+0].

Following the previous example, programs for 24 bit, 32 bit, 40 bit, etc. unsigned division can be written.

## About the Author

Name: M. Ganesh Raaja  
Title: R&D Engineer and owner of Omega Electronics & Consultants

Background: M. Ganesh does freelancing R&D work for various manufacturing industries. He obtained his diploma in Electronics and Communications Engg. in 1992. He has his own consultant firm called Omega Electronics & Consultants where they develop products for instrumentation and process control. He also develops and transfers technologies for manufacturing companies.  
Contact: 856, 9th Cross, Hebbal 2nd Stage, Mysore - 570 017, India.  
Nature of Work: Development of products for instrumentation and process control.  
Skill Set: Microcontrollers (assembly language programming), analog circuits, PCB designing using PROTEL.

**Table 1: Input/Output Details for the Subroutines**

Name	Input		Output	
	Dividend	Divisor	Result	Remainder
div8	[dividend+0]	[divisor+0]	[dividend+0]	[remainder+0]
div16	[dividend+1]	[divisor+1]	[dividend+1]	[remainder+1]
	[dividend+0]	[divisor+0]	[dividend+0]	[remainder+0]
div24	[dividend+2]	[divisor+2]	[dividend+2]	[remainder+2]
	[dividend+1]	[divisor+1]	[dividend+1]	[remainder+1]
	[dividend+0]	[divisor+0]	[dividend+0]	[remainder+0]

**Table 2: Code Size and Execution Time Details**

Name	Description	Code Size	Execution Time	
			Minimum	Maximum
div8	8-Bit Unsigned Division	39 Bytes	548 CPU Cycles	660 CPU Cycles
div16	16-Bit Unsigned Division	58 Bytes	1684 CPU Cycles	2100 CPU Cycles
div24	24-Bit Unsigned Division	77 Bytes	3452 CPU Cycles	4316 CPU Cycles

Cypress MicroSystems, Inc.  
 22027 17th Avenue S.E. Suite 201  
 Bothell, WA 98021  
 Phone: 877.751.6100  
 Fax: 425.939.0999

<http://www.cypressmicro.com/> / [http://www.cypress.com/aboutus/sales\\_locations.cfm](http://www.cypress.com/aboutus/sales_locations.cfm) / [support@cypressmicro.com](mailto:support@cypressmicro.com)

Copyright © 2002 Cypress MicroSystems, Inc. All rights reserved.

PSoC™ (Programmable System on Chip) is a trademark of Cypress MicroSystems, Inc.

All other trademarks or registered trademarks referenced herein are property of the respective corporations.

The information contained herein is subject to change without notice.