Unsigned Division Routines in PSoC® 1 Assembler

AN2101

Author: M. Ganesh Raaja

Associated Project: Yes
Associated Part Family: All PSoC® 1 Devices
Software Version: PSoC Designer™ 5.1
Associated Application Notes: AN2038

Application Note Abstract

Unsigned integer division is useful in many embedded applications, but is not available as a single CPU operation in all microcontrollers. In microcontrollers without a division instruction, division must be implemented as a series of assembly instructions. The following Application Note describes how to perform unsigned division for variables of any word length in PSoC® 1 devices.

Introduction

The M8C instruction set does not contain a division operation. 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. However, performing division in assembly is not as easy. To perform division in PSoC® 1, binary division algorithms must be used.

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

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.
- The results of the division are the quotient and the remainder.
- One more register, Z, is made an extension of register X such that the length of Z and X combined is twice that of X.
- At the end of division, X will contain the quotient with Z containing the remainder.

Steps Involved

1. Shift the double register (Z and X) to the left by 1.
2. Store a copy 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

Shown in Code 1 is a program that implements the above algorithm to perform an 8-bit unsigned division: This code is also demonstrated in the attached project.

Code 1. 8-Bit Unsigned Division in PSoC 1 Assembler

```
div8:

        mov [remainder+0],00h         ;initialize remainder to 0
        and F,fbh                             ;clear carry bit in flags
        mov [lcount],8                ;set loop counter to 8

d8u_1:
        rlc [dividend+0]                      ;shift MSB of dividend to LSB of remainder
        rlc [remainder+0]                     ; continued

        mov [temp+0],[remainder+0]    ;store remainder
        mov a,[remainder+0]           ;subtract divisor from remainder
        sub a,[divisor+0]                     ; continued
        mov [remainder+0],a           ; continued
        jnc d8u_2                             ;jump if result was positive

        mov [remainder+0],[temp+0]    ;restore remainder
        and [dividend+0],feh          ;clear LSB of dividend
        jmp chkLcount8                ;jump to loop counter decrement

d8u_2:
        or [dividend+0],01h           ;set dividend LSB to 1

chkLcount8:
        dec [lcount]                  ;decrement loop counter
        jnz d8u_1                             ;repeat steps if loop counter not zero
        ;division complete
```

Program Analysis

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

In lines fourth and fifth of the code, the dividend and remainder is rotated one bit to the left, i.e., the MSB of the dividend is shifted to the LSB of the remainder.

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

In lines seven, eight and nine, the divisor is subtracted from the remainder.

In line nine, 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]`.

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

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] [dividend+0]	[divisor+1] [divisor+0]	[dividend+1] [dividend+0]	[remainder+1] [remainder+0]
div24	[dividend+2] [dividend+1] [dividend+0]	[divisor+2] [divisor+1] [divisor+0]	[dividend+2] [dividend+1] [dividend+0]	[remainder+2] [remainder+1] [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

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 Eng. 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.

Document History

Document Title: Unsigned Division Routines in PSoC® 1 Assembler – AN2101

Document Number: 001-40479

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1532004	MAXK	10/02/2007	New Spec.
*A	3320543	MAXK	19/07/2011	Added project. Updated for newer PSoC 1 parts. Updated title.

In March of 2007, Cypress recataloged all of its Application Notes using a new documentation number and revision code. This new documentation number and revision code (001-xxxxx, beginning with rev. **), located in the footer of the document, will be used in all subsequent revisions.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com/