CSE 379

University at Buffalo

Keming Kuang

UBIT - kemingku

February 20, 2018

Contents

1 Description

As a low level language, ARM environment required working within low level communication between machine and human. In the lab2, we work on our own and develop a division function that can be able to solve unsigned division, signed division and calculate remainder.

1.1 Objective

Our objective is to write div and mod in ARM assembly language, in which the routine will take signed divisor and dividend, and return a cooresponding signed quotient and an unsigned remainder.

1.2 Usage

To run the div and mod routine, int divisor and int dividend in the following C language codes are supposed to be given with signed number.

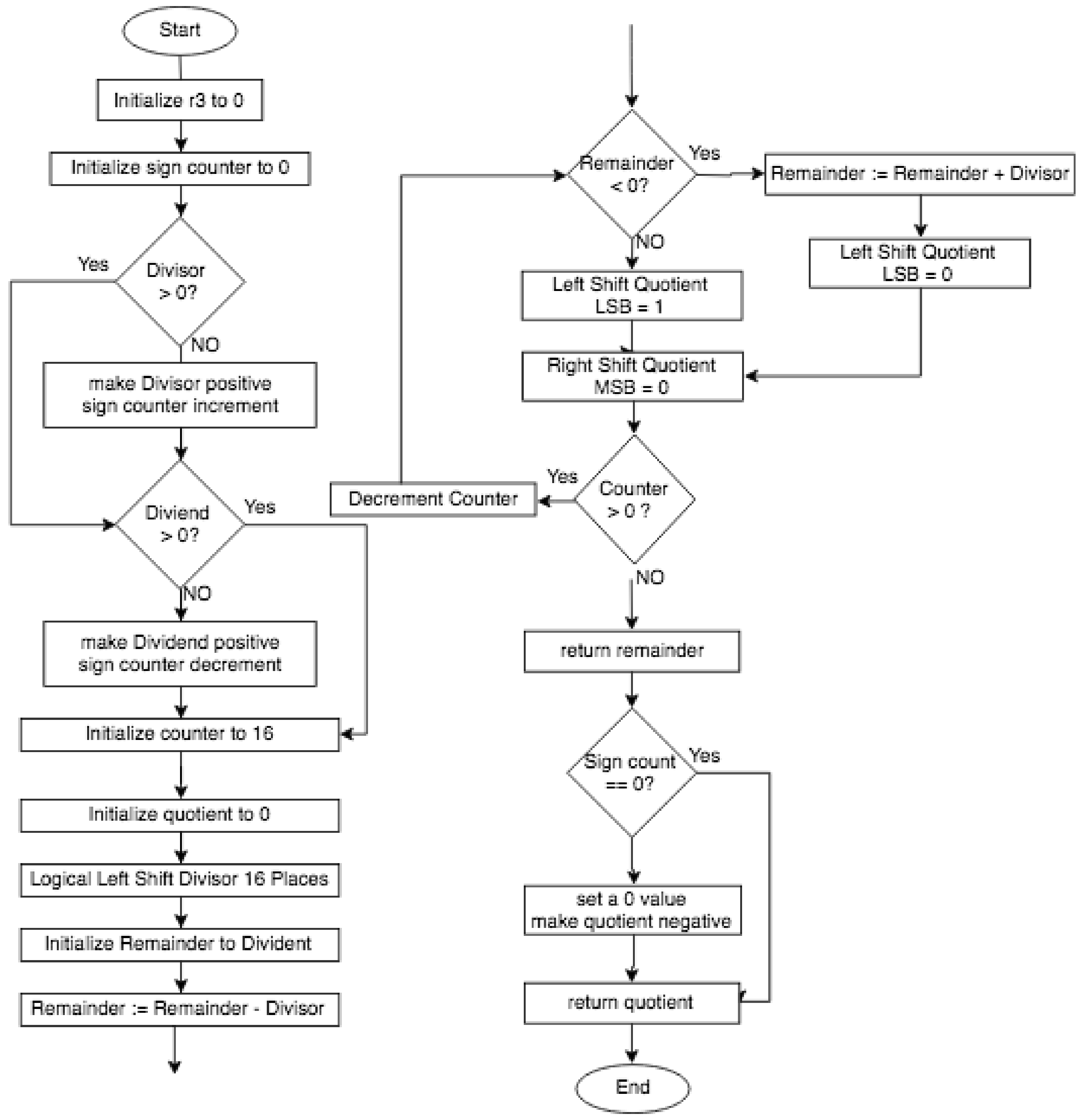
```c
extern int div_and_mod(int divisor, int dividend);

int main()
  {
  div_and_mod(21, -4718);
  }
```

1.3 Debugging

Before debugging, setup uVision to use memory layout from target dialogue to avoid read/write memory errors. Go to Options for Target in Project, and select Linker tab and Use Memory Layout from Target Dialogue. While debugging, The dividend is passed in r1 and the divisor in r0. After passing the value to div and mod routine, r1 will return a signed result and r0 will return the remainder. Be aware do not input r1 as 0.

2 Flow Chart


Start
Initialize r3 to 0
Initialize sign counter to 0
Divisor > 0?
Yes
NO
make Divisor positive
sign counter increment
Diviend > 0?
Yes
NO
make Dividend positive
sign counter decrement
Initialize counter to 16
Initialize quotient to 0
Logical Left Shift Divisor 16 Places
Initialize Remainder to Divident
Remainder := Remainder - Divisor
Remainder < 0?
Yes
NO
Remainder := Remainder + Divisor
Left Shift Quotient LSB = 0
Left Shift Quotient LSB = 1
Right Shift Quotient MSB = 0
Counter > 0 ?
Yes
NO
Decrement Counter
return remainder
Sign count == 0?
Yes
set a 0 value
make quotient negative
return quotient
End

3 Summary

To develop the div and mod routine, the logic behind it can be divided into 2 : unsigned division, signed division. The logic behind unsigned division is instead of direct subtraction, we use a more efficient approach to model an algorithm after long division. By subtracting the dividend by the largest (divisor multiply 2 to the power of n) we can get, in which base on the certain cumulative order that n from large to small, which is divisor shift right, as long as the remainder is larger than divisor, quotient will shift left and increment. on the other hand, the previous remainder will and quotient will shift left if the remainder is reached to negative value. For the signed division, both divisor and dividend will be determined if negative by compare and branch, and they will be set as positive for later calculation just the same as unsigned division. Set r5==0 as the sign count, when divisor is negative, r5 decrement. When dividend is negative, r5 increment. So only when there's one of divisor and dividend is negative, r5 will not equal to 0. Therefore, at the end of the code, only when r5 is not equal to 0, the signed quotient will be negative.