SPRING 2016

CMPE 364

Microprocessor Based Design

Dr. Mohamed Al-Meer and
Dr. Ryan Riley

Optimized Primitives

- A *primitive* is a basic operation that can be used in a wide variety of different algorithms and programs
  - For example, **addition**, **multiplication**, **division**, and **random number generation** are all primitives.
- Many primitives are not supported directly by instructions, and we **must write routines** to implement them
  - for example, **division** and **random number** generation, **Sin**, **Cos**, **Log**, and **square roots**.

Division in ARM

- We need access to very optimized division routines different from repeated subtraction method.
- Here we describe the fastest division implementations that we know.

- Suppose we need to calculate the quotient $q = n/d$ and remainder $r = n \% d$ for unsigned integers n and d.

Division by Subtraction

- Simplest (and slowest) algorithm.
- Keep subtracting d from n until we can't anymore.
- Example: 100/11

  100 – 11 = 89 (q=1)
  89 – 11 = 78 (q=2)
  78 – 11 = 67 (q=3)
  …
  12 – 11 = 1 (q = 9)
  Remainder is 1

Unsigned 32-Bit/32-Bit Divide by Trial Subtraction

- Quotient *q* fits into *N* bits so that $n/d < 2^N$, or equivalently $n < (d << N)$.
- calculates the *N* bits of *q* by trying to set each bit in turn, starting at the most significant bit, bit $N - 1$.
- We can set bit *k* if we can subtract $(d << k)$ from the current remainder without giving a negative result.
- Next will be the C code and next the ARM Assembly code

Unsigned 32-Bit/32-Bit Divide by Trial Subtraction – Example-1

- **Example – 1** (80 / 23)

$23 \times 2^7 > 80$	$23 \times 2^1 < 80$	$q = 2^1$.
$23 \times 2^6 > 80$		$80 - 23 \times 2^6 = 34$
$23 \times 2^5 > 80$	$23 \times 2^0 <$ **34**	$q \mathrel{+}= 2^0 = 3$
$23 \times 2^4 > 80$		
$23 \times 2^3 > 80$	r = 34 - 23 = 11	
$23 \times 2^2 > 80$	**80 / 23 = 3 // 11**	

Unsigned 32-Bit/32-Bit Divide by Trial Subtraction – Example-2

- **Example – 2** (200 / 3)

  $3 \times 2^7 > \mathbf{200}$

  $3 \times 2^6 < 200$ **q** = 2^6, **r** = 200 – $2^6 \times 3$ = 200 – 192 = **8**

  $3 \times 2^5 > \mathbf{8}$

  $3 \times 2^4 > 8$

  $3 \times 2^3 > 8$

  $3 \times 2^2 > 8$

  $3 \times 2^1 < 8$ **r** = 8 – 3×2^1 = **2**, **q** = q + 2^1 = 64 + 2 = **66**

  $3 \times 2^0 > 2$ **200 / 3 = 66 // 2**.

Unsigned 32-Bit/32-Bit Divide by Trial Subtraction

```
unsigned udiv_simple(unsigned d, unsigned n, unsigned N)
{
  unsigned q=0, r=n;

  do
  {                          /* calculate next quotient bit */
      N--;                   /* move to next bit */
      if ( (r>>N) >= d )     /* if r>=d*(1<<N) */
      {
        r -= (d<<N);         /* update remainder */
        q += (1<<N);         /* update quotient */
      }
    } while (N);

    return q;
  }
```

- See **C code** for Trial Subtraction.
- See next **Assembly** code as well.

Unsigned 32-Bit/32-Bit Divide by Trial Subtraction

```
mov n, #200
mov d, #3
mov bits, #8
mov q, #0
mov one, #1
mov r, n

loop:
       sub bits, bits, #1
       cmp d, r, LSR bits
       suble r, r, d, LSL bits
       addle q, q, one, LSL bits
       cmp bits, #0
       bne loop

bx lr
```

Questions, Problems and Discussions

How many steps to divide 130 / 12 using:

- **Normal subtraction**
- **Trial Subtraction**