


Firmware Division

Prof. Alberto Borghese
Dipartimento di Informatica
borghese@di.unimi.it

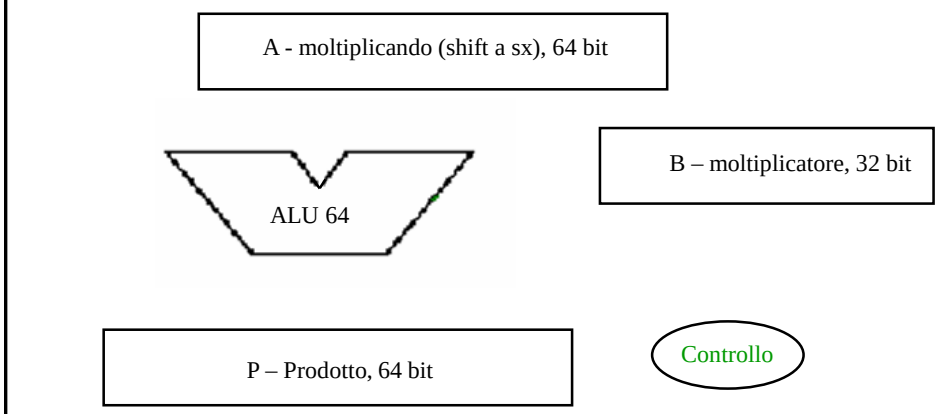
Università degli Studi di Milano

Riferimenti sul Patterson 5a ed.: 3.4, 3.5


Sommario

- Divisione intera
- Circuiti divisione intera
- Divisione e moltiplicazione

Implementazione circuitale
gli stessi attori della moltiplicazione



Algoritmi per la moltiplicazione

Il razionale degli algoritmi firmware della moltiplicazione è il seguente.

Si analizzano sequenzialmente i bit del moltiplicatore e:

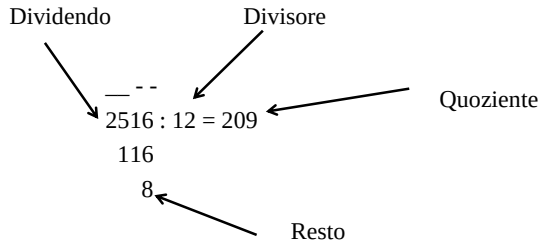
1) Si mette 0 nella posizione opportuna (se il bit analizzato del moltiplicatore = 0).

2) Si mette una copia del moltiplicando nella posizione opportuna (se il bit analizzato del moltiplicatore è = 1).

```
Moltiplicando      1 1 0 1 1 x
Moltiplicatore     1 0 1 =
                   ---------------------
                       1 1 0 1 1 +
                   → 0 0 0 0 0 -
                   ---------------------
                       1 1 0 1 1
                   → 1 1 0 1 1 - -
                   ---------------------
Prodotto           1 0 0 0 0 1 1 1
```

La moltiplicazione viene effettuata come somme successive, con peso crescente, di uno tra i 2 valori: {moltiplicando, 0}

La divisione decimale



Dividendo = Divisore * Quoziente (quoto) + Resto

Razionale della divisione decimale

la divisione è l'operazione inversa del prodotto

2628 : 12 = 219 (resto = 0) ➔ 219 x 12 (+ resto) = 2628

(2x100) x 12 + (1x10) x 12 + (9x1) x 12 = 2628

Da cui segue che ad ogni passo della divisione erodiamo una quantità via via decrescente. Al primo passo:

(1x10) x 12 + (9x1) x 12 = 2628 – (2x100) x 12 = 2628 – 2400 = 228

Abbiamo cioè eroso 12 centinaia. Al passo successive eroderemo in questo caso 1 decina e al passo finale 9 unità. Quello che rimane è il resto.

La divisione decimale

```
    - - - -
    2516 : 12 = 0209
12x0 = 0 =
    ----
    25 -
12x2 = 24=                    Erodiamo 2 centianaia di 12
    -----
     11-
12x0 =  0=                    Erodiamo 0 decine di 12
    -------
12x9 =  116 -
        108 =                 Erodiamo 9 unità di 12
        -------
           8      Resto
```

Dividendo = Divisore * Quoziente (quoto) + Resto

La divisione decimale::algoritmo

```
      _ _ _ _
Passo 1)  2 5 1 6 : 0012 = 0209     Il 12 nel 2 ci sta 0 volte.
         -0                         12 x 0 = 0
         --
          2                         Resto parziale - I
Passo 2)  2 5                       Considero il 5 del divisore e lo affianco al resto parziale. Il
                                    12 nel 25 ci sta 2 volte.
         -2 4                       12 x 2 = 24
         -----
            1                       Resto parziale - II
Passo 3)    1 1                     Considero l'1 del dividendo e lo affianco al resto parziale.
                                    Il 12 nell'11 ci sta 0 volte.
             -0                     12 x 0 = 0
            -----
            1 1                     Resto parziale – III
Passo 4)    1 1 6                   Considero il 6 del dividendo e lo affiunco al resto parziale.
                                    Il 12 nel 16 ci sta 1 volta.
           -1 0 8                   12 x 9 = 108
           ------
              8                     Resto parziale – IV = RESTO
```

La divisione tra numeri binari

Divisione decimale fra i numeri a = 1.001.010 e b = 1.000 a : b = ?

```
____
1001010 : 1000 = 1          Dividendo : Divisore        74 : 8 = 9  resto 2
1000-
-----
   1


____ - - -
1001010 : 1000 = 1001       Quoziente
1000-
-----
   1010  Resto parziale
   1000-
   -----
     10  Resto
```

Dividendo = Quoziente x Divisore + Resto

Confronto tra moltiplicazione e divisione

- La moltiplicazione opera per somme successive di quantità pesate con peso crescente.
- La sottrazione opera erodendo dal dividendo quantità pesate con peso decrescente.
- La moltiplicazione è costituita da somme ripetute del moltiplicando.
- La divisione è costituita da sottrazioni ripetute (del divisore)
  - N : M = Q + R => N - M * Q = R

Confronto tra divisione tra numeri binari e decimali

In DECIMALE:

Ad ogni passo devo verificare QUANTE VOLTE il resto parziale contiene il divisore. Il risultato è un numero che va da 0 a 9 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. 0 = il divisore non è contenuto nel resto.

In BINARIO:

Ad ogni passo devo verificare SE il resto parziale contiene il divisore. Ovverosia se lo contiene 0 o 1 volta. Il risultato è un numero che può valere {0, 1}.

In DECIMALE:

Il numero che viene sottratto al resto parziale è ottenuto moltiplicando il divisore per una delle cifre da 0 a 9.

In BINARIO:

Il numero che viene sottratto al resto parziale può essere solamente 0 o il DIVISORE stesso

In entrambi i casi il quoziente si forma dalla cifra più significativa, cioè da sinistra a destra.

Peculiarità della divisione

- Come rappresento la condizione “il divisore è contenuto nel resto parziale”?

Esempi:

10 (resto parziale) non è contenuto in 1000 (divisore)

1001 (resto parziale è contenuto in 1000 (divisore)

Per tentativo.

Eseguo la sottrazione.

Risultato >= 0 → il resto parziale è contenuto nel divisore.

Risultato < 0 → il resto parziale non è contenuto nel divisore (è più piccolo del divisore).

Osserviamo che nel secondo caso abbiamo fatto in realtà una sottrazione che non avremmo dovuto effettuare.

NB il calcolatore non può sapere se il resto parziale contiene il divisore fino a quando non ha effettuato la sottrazione.

La divisione tra numeri binari

Divisione decimale fra i numeri su 7 bit: a = 100 1010 e b = 000 1000 a : b = ? 74 : 8=?

Nel primo passaggio allineo il divisore con la prima cifra significativa (=1) del dividendo.

Allocazione di 14 bit

0000 000 100 1010 - 0000 100 000 0000 =

1111100100 1010

0000 000 100 1010 + 1111 100 000 0000 =

1111 100 100 1010 (= $-438|_{10}$)

Resto parziale = dividendo - divisore * 2^6

Nuovo resto parziale provvisorio

Ripristino il resto parziale precedente: 0000 000 100 1010

Note e strategia di implementazione

I bit del dividendo vengono analizzati da sx a dx. Il divisore viene spostato verso dx di 1 bit ad ogni passo.

Il quoziente cresce dal bit più significativo verso il bit meno significativo. Cresce verso dx.

All'inizio il divisore è allineato alla sinistra del dividendo: le cifre del dividendo sono allineate agli 0 del divisore e gli 0 del dividendo sono allineati alle cifre del divisore.

Ci sono N+1 passi di divisione, il primo darà sempre 0 e si potrebbe omettere.

Occorre quindi effettuare:

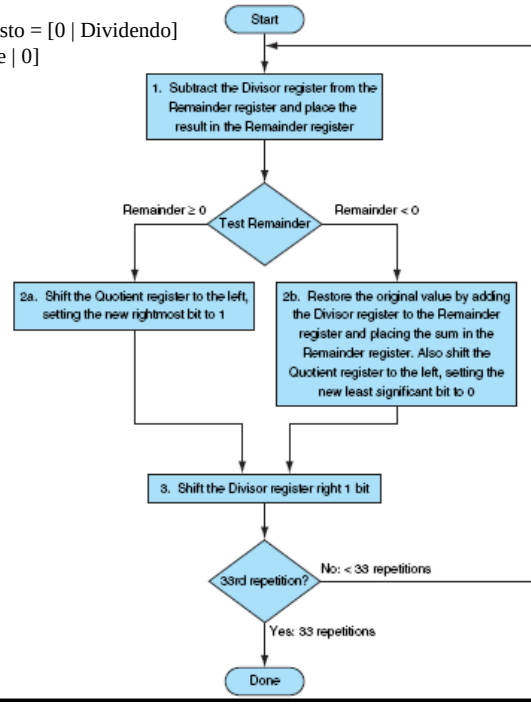
- Shift quoziente a sx ad ogni passo.
- Scrittura di 1 o 0 nel registro quoziente.
- Shift del divisore verso dx ad ogni passo..

Utilizzo un unico registro per dividendo e resto.

Considero il primo resto parziale uguale al dividendo (inizializzazione).

Il primo passo sarà "a vuoto" perchè produrrà sempre quoziente 0.

Inizializzazione: Resto = [0 | Dividendo]
Divisore = [Divisore | 0]
Quoziente = 0
k = 0



Divisione:: algoritmo per 32 bit

Esempio

Divisione decimale fra i numeri a = 7 e b = 2 a : b = ?

Inizializzo il divisore alla sinistra delle quattro cifre significative. La prima cifra del quoziente sarà sempre 0.

Iteration	Step	Quotient	Divisor	Remainder
0	Initial values	0000	0010 0000	0000 0111
1	1: Rem = Rem – Div	0000	0010 0000	①110 0111
	2b: Rem < 0 ⟹ +Div, sll Q, Q0 = 0	0000	0010 0000	0000 0111
	3: Shift Div right	0000	0001 0000	0000 0111
2	1: Rem = Rem – Div	0000	0001 0000	①111 0111
	2b: Rem < 0 ⟹ +Div, sll Q, Q0 = 0	0000	0001 0000	0000 0111
	3: Shift Div right	0000	0000 1000	0000 0111
3	1: Rem = Rem – Div	0000	0000 1000	①111 1111
	2b: Rem < 0 ⟹ +Div, sll Q, Q0 = 0	0000	0000 1000	0000 0111
	3: Shift Div right	0000	0000 0100	0000 0111
4	1: Rem = Rem – Div	0000	0000 0100	⓪000 0011
	2a: Rem ≥ 0 ⟹ sll Q, Q0 = 1	0001	0000 0100	0000 0011
	3: Shift Div right	0001	0000 0010	0000 0011
5	1: Rem = Rem – Div	0001	0000 0010	⓪000 0001
	2a: Rem ≥ 0 ⟹ sll Q, Q0 = 1	0011	0000 0010	0000 0001
	3: Shift Div right	0011	0000 0001	0000 0001

Esempio – step 1

Il resto parziale è inizializzato a: [0 | dividendo]: 0000 0111

```
  __
 0000 0111 : 0010 =
-0010 0000
--------
   <0
```

Il divisore non è contenuto nel resto parziale

Osservando i registri, in pratica, eseguiamo la differenza tra:

```
0000 0111 –        7          - Resto parziale
0010 0000 =       32 = 2*2^4  - Divisore allineato al di fuori e alla sx dei 4 bit del dividendo.
-------------
1110 0111+        -25_10      < 0
0010 0000=
-------------
0000 0111          +7
```

Quoziente al passo 1: 0
Storno la sottrazione

Esempio – step 2

Il resto parziale è ancora uguale al dividendo: 0000 0111

```
  __
 0000 0111 : 0010 =
-0001 0000
---------------
   <0
```

Il divisore (0001 0000) non è contenuto nel resto parziale

Osservando i registri, in pratica, eseguiamo la differenza tra:

```
0000 0111 –        7          - Resto parziale
0001 0000 =       16 = 2*2^3  - Divisore allineato al MSB dei 4 bit del dividendo.
-------------
1111 0111+        -9_10 +     < 0
0001 0000=        16
-------------
0000 0111          +7
```

Quoziente al passo 2: 00
Storno ancora la sottrazione

Esempio – step 3

Il resto parziale è ancora uguale al dividendo: 0000 0111

```
    __
 0000 0111 : 0010 =
-0000 1000
  -----------
   <0
```

Il divisore (0000 1000) non è contenuto nel resto parziale (0111)

Osservando i registri, in pratica, eseguiamo la differenza tra:

0000 0111 –	7	- Resto parziale
0000 1000 =	$8 = 2 * 2^2$	- Divisore allineato al terzo dei 4 bit del dividendo.
1111 1111+	-1_{10}	< 0
0000 1000=	+8	
0000 0111	+7	

Quoziente al passo 3: 000
Storno ancora la sottrazione

Esempio – step 4

Il resto parziale è ancora uguale al dividendo: 0000 0111

```
    __
 0000 0111 : 0010 =
-0000 0100
  --------
   0 001          > 0
```

Il divisore (0000 0010) **è** contenuto nel resto parziale

Osservando i registri, in pratica, eseguiamo la differenza tra:

0000 0111 –	Resto parziale	
0000 0100 =	Divisore allineato al secondo dei 4 bit del dividendo.	
0000 0011	3_{10}	E' il nuovo resto parziale $7 – 1*2^2 = 3$

Quoziente al passo 4: 0001
Non storno la sottrazione.
Il resto non è più uguale al dividendo.

Esempio – step 5

Il resto parziale = 0000 0011

```
    __
0000 0111 : 0010 =
-0000 0010
--------
 0001          > 0
```

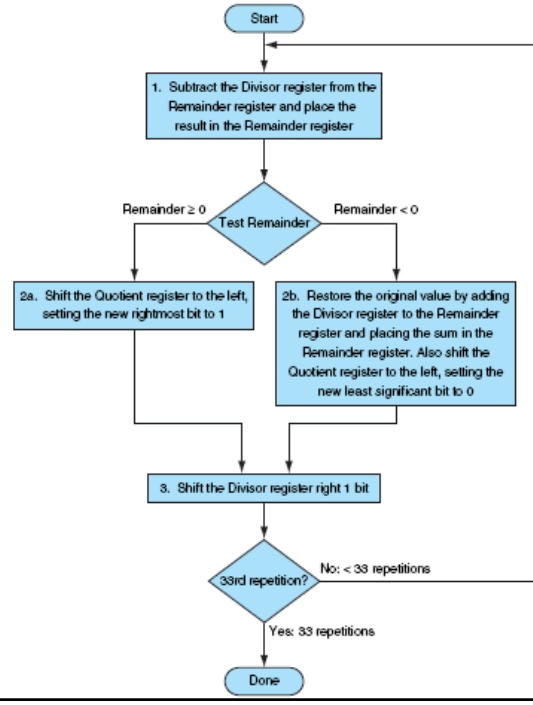
Il divisore (0010) **è** contenuto nel resto parziale

Osservando i registri, in pratica, eseguiamo la differenza tra:

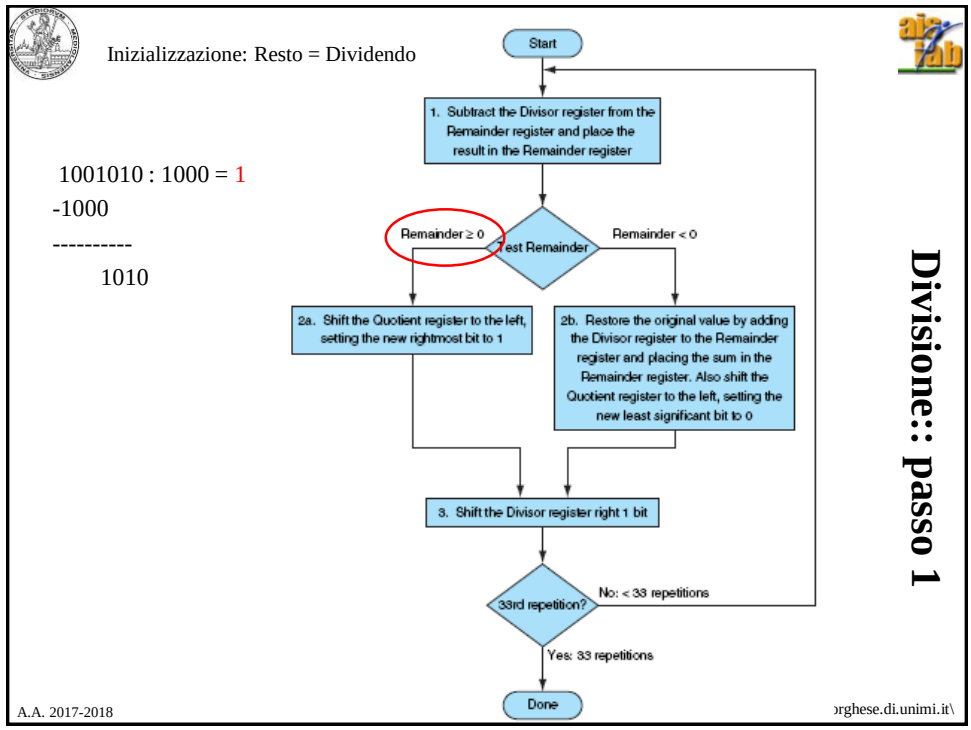
0000 0011 – Resto parziale
0000 0010 = Divisore allineato LSB dei 4 bit del dividendo.

0000 0001 1_{10} E' il nuovo resto parziale $3 - 1*2^1 = 1$ RESTO FINALE

Quoziente al passo 5: 00011



Divisione:: algoritmo

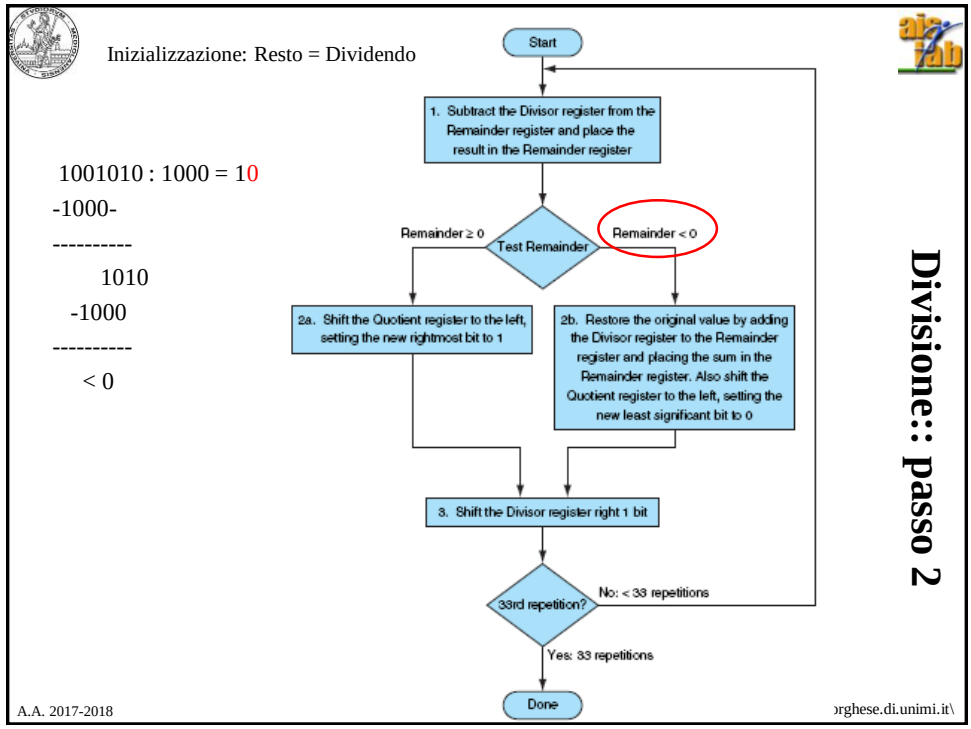


Inizializzazione: Resto = Dividendo

1001010 : 1000 = 1
-1000

1010

Divisione:: passo 1



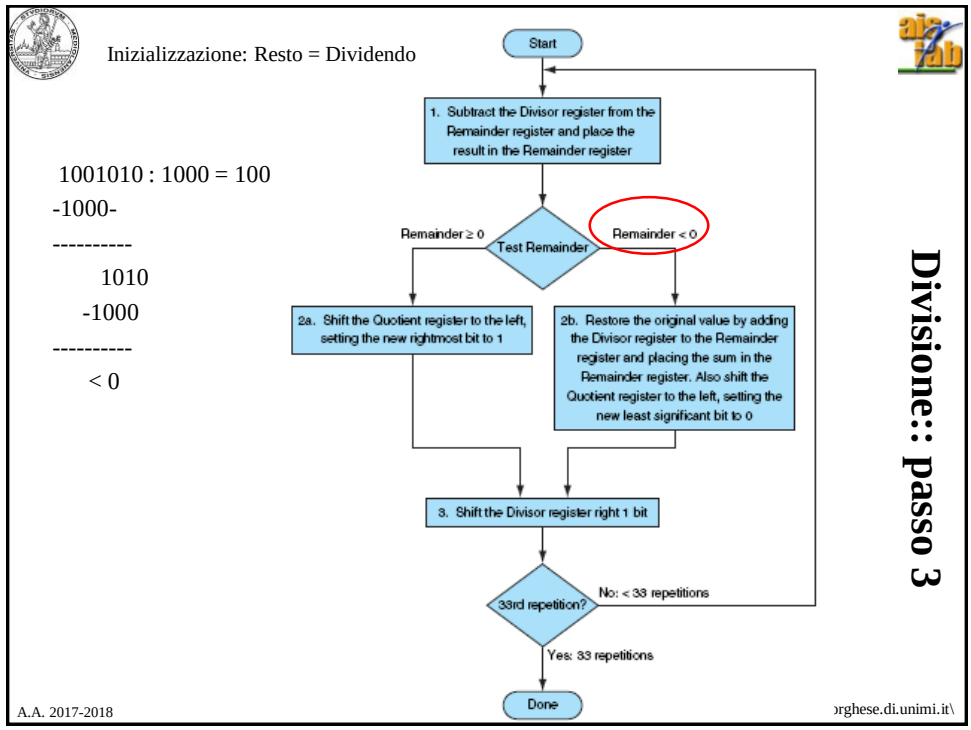
Inizializzazione: Resto = Dividendo

1001010 : 1000 = 10
-1000-

1010
-1000

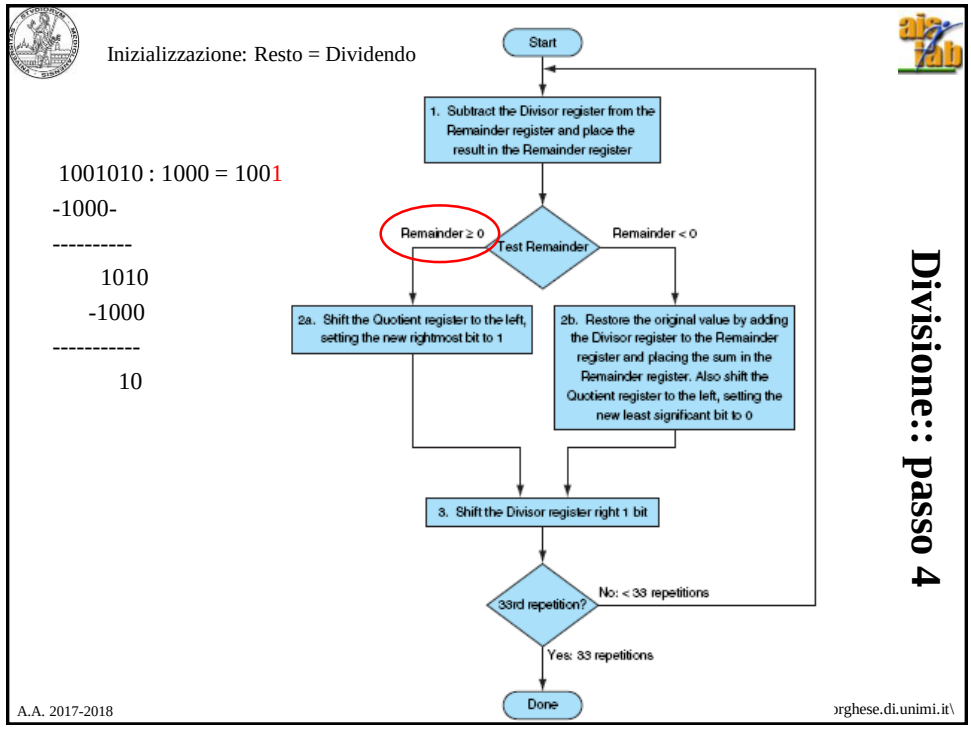
< 0

Divisione:: passo 2


Inizializzazione: Resto = Dividendo
1001010 : 1000 = 100
-1000-

1010
-1000

< 0
Start
1. Subtract the Divisor register from the Remainder register and place the result in the Remainder register
Test Remainder
Remainder ≥ 0
Remainder < 0
2a. Shift the Quotient register to the left, setting the new rightmost bit to 1
2b. Restore the original value by adding the Divisor register to the Remainder register and placing the sum in the Remainder register. Also shift the Quotient register to the left, setting the new least significant bit to 0
3. Shift the Divisor register right 1 bit
33rd repetition?
No: < 33 repetitions
Yes: 33 repetitions
Done
Divisione:: passo 3
A.A. 2017-2018
orghese.di.unimi.it\


Inizializzazione: Resto = Dividendo
1001010 : 1000 = 1001
-1000-

1010
-1000

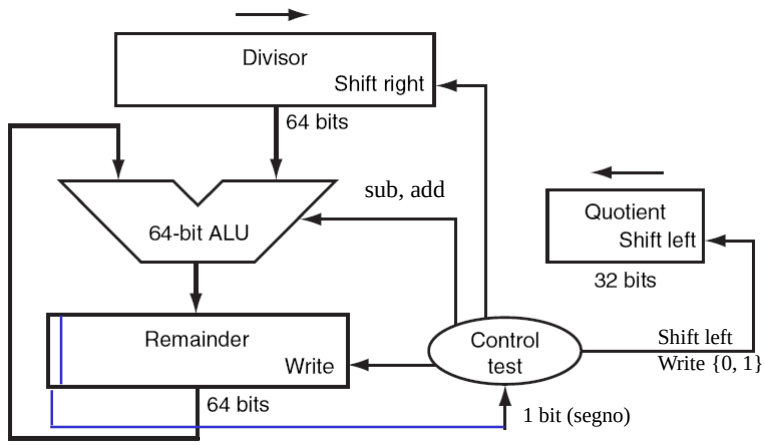
10
Start
1. Subtract the Divisor register from the Remainder register and place the result in the Remainder register
Test Remainder
Remainder ≥ 0
Remainder < 0
2a. Shift the Quotient register to the left, setting the new rightmost bit to 1
2b. Restore the original value by adding the Divisor register to the Remainder register and placing the sum in the Remainder register. Also shift the Quotient register to the left, setting the new least significant bit to 0
3. Shift the Divisor register right 1 bit
33rd repetition?
No: < 33 repetitions
Yes: 33 repetitions
Done
Divisione:: passo 4
A.A. 2017-2018
orghese.di.unimi.it\

Sommario

- Divisione intera
- Circuiti divisione intera
- Divisione e moltiplicazione

Il circuito firmware della divisione

Inizializzazione: Resto = 0 | Dividendo



Come ottimizzare il circuito della divisione

Il resto si sposta di 1 bit alla volta verso dx ma rimane pari al numero di bit della parola.
Possiamo evitare di spostare il resto.

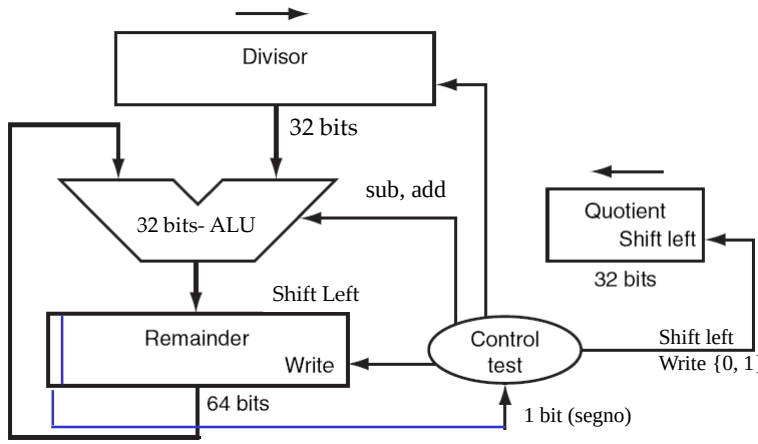
Il divisore si sposta verso dx di un bit ad ogni passo e viene sottratto al resto parziale.
Otteniamo lo stesso risultato se spostiamo il resto parziale a sx di un bit ad ogni passo.
Il quoziente si sposta verso sx ad ogni passo.

Inizializziamo il resto come RESTO = 0 | DIVIDENDO.
Alla fine avremo: RESTO | QUOZIENTE.

Ad ogni passo sposto il dividendo di una posizione a sx ed inserisco un bit del quoziente.

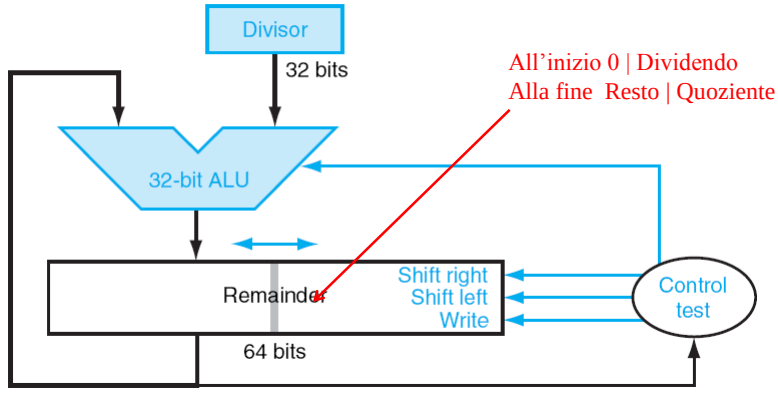
Il circuito firmware con un'ottimizzazione

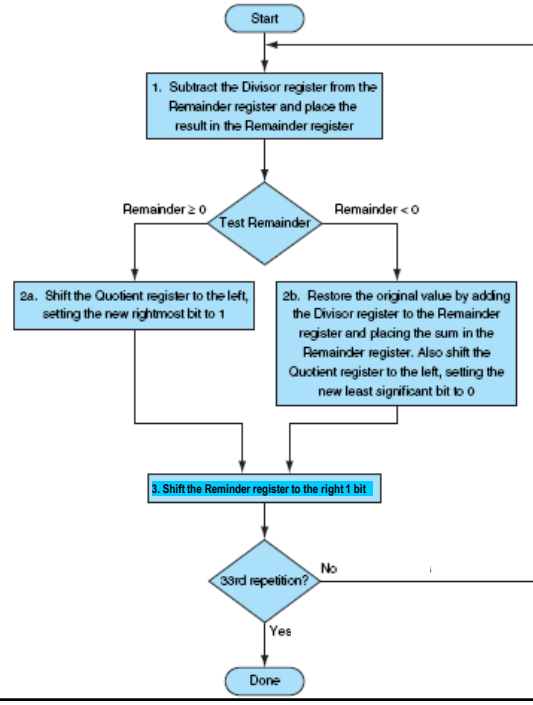
Inizializzazione: Resto = 0 | Dividendo



Il circuito firmware ottimizzato della divisione

Inizializzazione: Resto = 0 | Dividendo



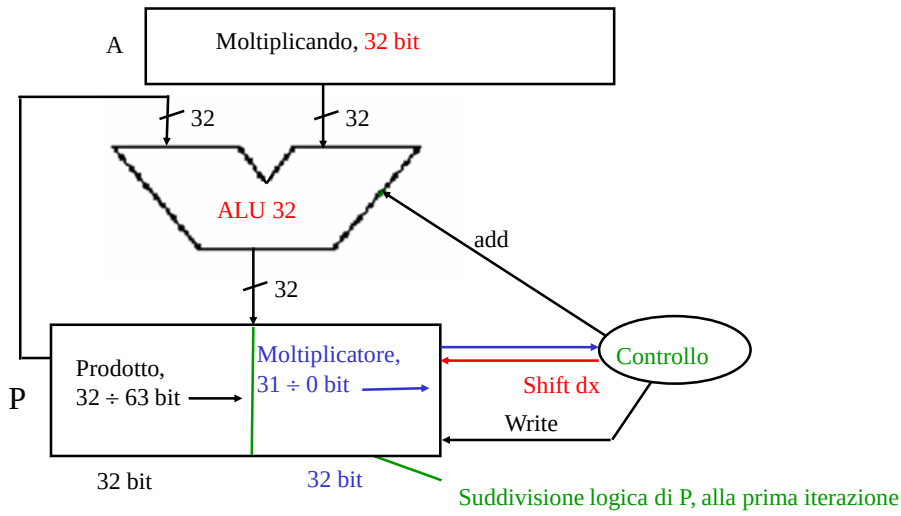


Divisione:: algoritmo

Sommario

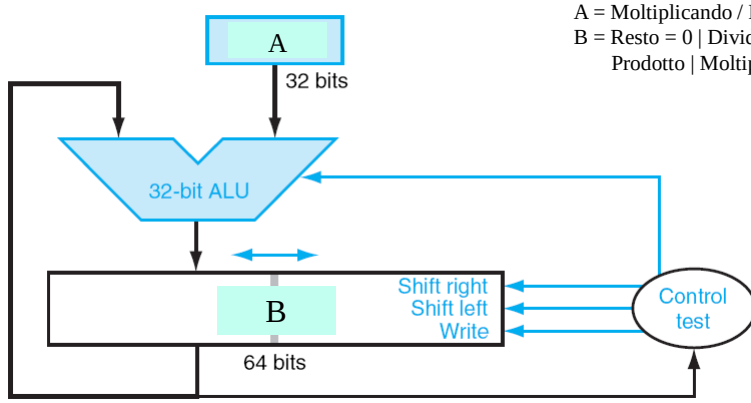
- Divisione intera
- Circuiti divisione intera
- Divisione e moltiplicazione

Circuito ottimizzato della moltiplicazione



Un unico circuito per moltiplicazione e divisione

Inizializzazione:
A = Moltiplicando / Divisore
B = Resto = 0 | Dividendo /
Prodotto | Moltiplicatore



Sommario

- Divisione intera
- Circuiti divisione intera
- Divisione e moltiplicazione