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

Moltiplicazione

128 x		Dividendo		
214 =		Divisore		
--------				
512 +	$x10^0$	Prodotto parziale	$128x(4x10^0)+$	unità
128- +	$x10^1$	Prodotto parziale	$128x(1x10^1)+$	decine
256-- =	$x10^2$	Prodotto parziale	$128x(2x10^2) =$	centinaia

27392				

Prodotto come **somma dei prodotti parziali**. Ognuno dei prodotti parziali aggiunge il **moltiplicando pesato con il peso della cifra analizzata del moltiplicatore** x un numero di volte pari alla cifra analizzata del moltiplicatore.

Nel caso binario le cifre sono solo 0,1 ===➔

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

```
__ - -
3716 : 12 = 309
36--
-------
  11-
   0-
------
  116
  108
  -----
    8
```

Resto

Dividendo = Divisore * Quoziente (quoto) + Resto

Sviluppo della divisione - I


```
__ - -
3716 : 12 = 309
36--
-------
  11-
   0-
------
  116
  108
  -----
    8
```


1. Analizzo il dividendo

2. Identifico quale potenza di 10 iniziare a considerare. Questa è la **massima potenza, p,** tale per cui 10^p moltiplicata per il divisore mi dà un valore inferiore al divendo.

In questo caso:

p = 1 –> 10 * 12 = 120　　≤ 3716
p = 2 -> 100 * 12 = 1200　　≤ 3716
p = 3 –> 1000*12 = 12000　　> 3716

E' necessario considerare tutte le cifre del dividendo? No.

In questo caso:

p = 1 –> 10 * 12 = 120　　≤ 3700
p = 2 -> 100 * 12 = 1200　　≤ 3700
p = 3 –> 1000*12 = 12000　　> 3700

Si considerano solo le cifre associate alla potenza che va da 10^p in poi: abbiamo una risoluzione pari a 10^p

Sviluppo della divisione - II

```
__ - -
3716 : 12 = 309
36--
-------
 11-
  0-
------
 116
 108
-----
   8
```

3. Determino qual è il **numero massimo di volte, q_2,** per cui il divisore x 10^p può essere moltiplicato per ottenere un numero inferiore al dividendo.

In questo caso:

q_2 = 1 -> 1 * 1200 = 1200 ≤ 3700
q_2 = 2 -> 2 * 1200 = 2400 ≤ 3700
q_2 = 3 -> 3 * 1200 = 3600 ≤ 3700
q_2 = 4 -> 4 * 1200 = 4800 > 3700

NB la moltiplicazione x 10^p si sottoindende, allineando il q_2 del divisore alla cifra opportuna. In questo caso 3x12 = 36 viene allineato alle centinaia (p = 2).

Sviluppo della divisione - III

```
__ - -
3716 : 12 = 309
3600
-------
  116
0000
------
  116
  108
-----
    8
```

4. Ho stabilito p = 2, q_2 = 3 ➔ posso **erodere una quantità pari a:**

$3 * 12 * 10^2 = 3600$

3716 -	q_2 = 3 è la cifra del quoziente cioè di quanto stiamo erodendo
3600 =	p = 2 ➔ 10^2 è il peso del divisore

116	Resto parziale

I passi successivi al primo sono leggermente diversi:
- Invece del dividendo considero il resto parziale
- Decremento di 1 l'esponente della base, p.

Riassunto sulla divisione

1. Analizzo il dividendo
2. Identifico quale potenza di 10 iniziare a considerare. Questa è la **massima potenza, p,** tale per cui 10^p moltiplicata per il divisore mi dà un valore inferiore al divendo (si considerano solo le cifre del dividend da 10^p in poi). In questo caso:

   p = 1 –> 10 * 12 = 120 ≤ 3700
   p = 2 -> 100 * 12 = 1200 ≤ 3700
   p = 3 –> 1000*12 = 12000 > 3700

3. Determino qual è il **numero massimo di volte, q_2,** per cui il divisore x 10^p può essere moltiplicato per ottenere un numero inferiore al dividend (si considerano solo le cifra dalla posizione 10^p in poi). In questo caso:

   q_2 = 2 -> 2 * 1200 = 2400 ≤ 3700
   q_2 = 3 -> 3 * 1200 = 3600 ≤ 3700
   q_2 = 4 -> 4 * 1200 = 4800 > 3700

4. Ho stabilito p = 2, q_2 = 3 ➔ posso **erodere una quantità pari a: 3 * 12 * 10^2 = 3600**

   3716 - q_2 = 3 è la cifra del quoziente cioè di quanto stiamo erodendo
   3600 = p = 2 ➔ 10^2 è il peso del divisore
   -----
   116 Resto parziale

Procedo ricorsivamente ripetendo i passi 2-4 sul resto fino a quando non ottengo un resto inferiore al divisore. **Nei passi successive invece del dividendo si considera il resto parziale.**

Esempio della divisione - I

— - -
3716 : 12 = 309
36-- Centinaia

11-
0- Decine

116
108 Unità

8

1. Analizzo il dividendo

2. Considero la potenza max p = **2.**

   p = 1 –> 10 * 12 = 120 ≤ 3700
   p = 2 -> 100 * 12 = 1200 ≤ 3700
   p = 3 –> 1000*12 = 12000 > 3700

3. Determino qual è il **numero massimo di volte, q_2,** per cui il divisore x $10^{p=2}$ può essere moltiplicato per ottenere un numero inferiore al dividendo. (si considerano solo le cifre dalla posizione 10^2 in poi). In questo caso:

   q_2 = 2 -> 2 * 1200 = 2400 ≤ 3700
   q_2 = 3 -> 3 * 1200 = 3600 ≤ 3700
   q_2 = 4 -> 4 * 1200 = 4800 ≤ 3700

4. Ho stabilito p = 2, q_2 = 3 ➔ posso **erodere una quantità pari a 3 * 12 * 10^1 = 3600 ->**
   Primo resto parziale = 116

Esempio della divisione - II

3716 : 12 = 309
36--

11-
0-

116
108

8

1. Considero il primo resto parziale (116).

2. Considero la potenza p = **p-1 ->10^1.**
   p = 1 –> $10^1 * 12 = 120$

3. Determino qual è il **numero massimo di volte, q_1,** per cui il divisore x 10^p può essere moltiplicato per ottenere un numero inferiore al resto parziale (si considerano solo le cifre dalla posizione 10^1 in poi). In questo caso:
   $q_1 = 0 \rightarrow 0 * 120 = 0 \quad \leq 110$
   $q_1 = 1 \rightarrow 1 * 120 = 120 \quad > 110$

4. Ho stabilito p = 1, $q_1 = 0$ ➔ posso **erodere una quantità pari a: $0 * 12 * 10^1 = 0$**

116 | $q_1 = 0$ è la cifra del quoziente cioè di quanto stiamo erodendo
 00 = | p = 1 ➔ 10^1 è il peso del divisore
-----
116 | Secondo resto parziale

Esempio della divisione - III

3716 : 12 = 309
36--

11-
0-

116
108

8

1. Considero il secondo resto parziale (=116)

2. Considero la potenza p=**p-1, 0 in questo caso.**
   p = 0 –> 1 * 12 = 12

3. Determino qual è il **numero massimo di volte, q_0,** per cui il divisore x 10^p può essere moltiplicato per ottenere un numero inferiore al resto parziale (si considerano solo le cifre dalla posizione 10^p in poi). In questo caso:
   $q_0 = 8 \rightarrow 8 * 12 = 96 \quad < 116$
   $q_0 = 9 \rightarrow 9 * 12 = 108 \quad < 116$

4. Ho stabilito p = 0, $q_0 = 9$ ➔ posso **erodere una quantità pari a: $9 * 12 * 10^0 = 108$**

116 - | $q_0 = 9$ è la cifra del quoziente cioè di quanto stiamo erodendo
108 = | p = 0 ➔ 10^0 è il peso del divisore
----
  8 | Resto totale

Razionale della divisione decimale

La divisione è l'operazione inversa del prodotto

2629 : 12 = 219 **(resto = 1)** ➔ 219 x 12 + 1 = 2629

[**($2x10^2$)** + **($1x10^1$)** + **($9x10^0$)**] x 12 **+ 1** = 2629

Da cui segue che ad ogni passo della divisione erodiamo **(sottraiamo)** una quantità via via decrescente. Al primo passo sottraiamo a entrambi i membri:

1. [**($1x10^1$)** + **($9x10^0$)**] x 12 +1 = 2629 **– ($2x10^2$) x 12** = 2629 – 2400 = 229 -> resto parziale. Abbiamo cioè eroso 24 centinaia al dividendo 2629.

2. Al passo successivo eroderemo al resto parziale 12 decine:
[**($9x10^0$)**] x 12 + 1 = 229 – **($1x10^1$) x 12** = 229 – 120 = 109 -> resto parziale

3. E al passo finale eroderemo 12x9=108 unità:
[1 = 109 – **($9x10^0$) x 12** = 1 -> **resto finale.**

Ad ogni passo si cerca di erodere il più possibile!

Confronto con la moltiplicazione

La divisione è l'operazione inversa del prodotto

2629 : 12 = 219 **(resto = 1)** ➔ 219 x 12 + 1 = 2629

Divisione: Ad ogni passo erodiamo (sottraiamo) un certo numero di volte il divisore moltiplicato per la massima cifra possibile, moltiplicato anche per la potenza della base massima possible -> cifra del quoziente.

Opera per sottrazioni successive di quantità sempre più piccole proporzionali al divisore, producendo via via i resti parziali.

Moltiplicazione: a ogni passo sommiamo il moltiplicando moltiplicato per il numero di volte indicato dalla cifra del moltiplicatore considerata e per la potenza associata alla posizione della cifra del moltiplicatore, che stiamo considerando.

Opera per somme successive di quantità sempre più grandi proporzionali al moltiplicatore, producendo via via le somme parziali.

La divisione tra numeri binari

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

```
____
1001010 : 1000 = 1001              Dividendo : Divisore        74 : 8 = 9  resto 2
1000 -
-----
   10    -                    Quoziente
 0000    =
---------                     Resti parziali
   101   -
  0000   =
---------
  1010   –
  1000   =
----------
    10        Resto
```

Dividendo = Quoziente x Divisore + Resto

Tutti numeri binari

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}.

```
                                 ____
__ - -                           1001010 : 1000 = 1001
3716 : 12 = 309                  -----
3600                                1010   -
-------                           000000   =
 116                             ---------
  00                                1010   -
------                             00000   =
 116                             ---------
 108                                1010   –
-----                               1000   =
   8                             ----------
                                      10
```

Peculiarità della divisione

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

Esempio:
- 10 (resto parziale) non contiene 1000 (divisore)
- 1001 (resto parziale) contiene 1000 (divisore)

Procedo per tentativi.
- Eseguo la sottrazione.
  - Risultato >= 0 ➔ il resto parziale è maggiore del divisore.
  - Risultato < 0 → il resto parziale è minore del divisore (non contiene il 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.

Scelta automatica della potenza

Occorre definire l’ampiezza, N, della parola che termina il range degli attori.

E.g. N = 5 -> potenza massima = 10^5 -> Dividendo massimo sarà: 99 999

Per capire da quale potenza partire si incolonna il divisore alla potenza massima e poi si diminuisce a ogni passo:

Consideriamo la divisione: 99 999 : 3 = 33 333

Parto considerando 3 x 10^5 = 300000

Effettuo la sottrazione al passo 1

00000 **99999** -
0000**3 00000** =
-2 00001 < 0 ho eroso troppo

Effettuo la sottrazione al passo 2

00000 **99999** -
00000 **30000** =
69999 > 0 posso erodere,
determino q=3

Allineo il divisore alla potenza immediatamente successiva alla potenza massima del dividendo.

La divisione tra numeri binari – primo passo

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 alla sinistra della prima cifra (MSB) del dividendo.

Allocazione di **2N = 14 bit**. Sottrazione -> somma in complemento a 2:

000 0000 100 1010 -	000 0000 100 1010 +	Resto parziale = dividendo - divisore * 2^7
000 1000 000 0000 =	111 1000 000 0000 =	= 74–8*2^7 = 74-1024 =-950
-----------------------	------------------------	
111 1000 100 1010	111 1000 100 1010	(= $-950|_{10}$) Nuovo resto parziale provvisorio

Ripristino il resto parziale precedente: 0000 000 100 1010

Note e strategia di implementazione

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.

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.

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

Occorre quindi effettuare a ogni passo:

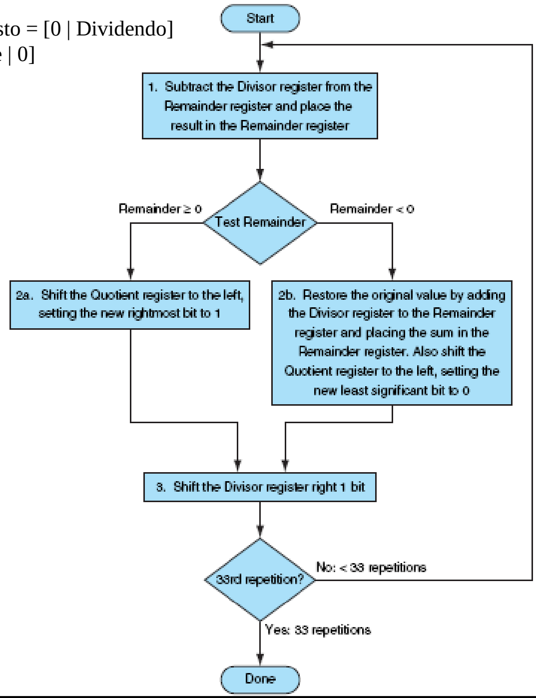
Resto parziale = Resto parziale – divisore * base corrente
Ripristino del resto parziale (se il risultato della sottrazione < 0)
Shift quoziente a sx.
Scrittura di 1 o 0 nel registro quoziente.
Shift del divisore verso dx.

- Utilizzo un unico registro per dividendo e resto (il resto si ottiene per erosione del dividendo).
- Considero il primo resto parziale uguale al dividendo (inizializzazione).
- Il primo passo sarà "a vuoto" perchè produrrà sempre quoziente 0.

Divisione:: algoritmo per 32 bit

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

```
_____
1001010 : 1000 = 1001
-----
    1010    -
  000000    =
 ---------
    1010   -
   00000  =
 ---------
    1010 –
    1000 =
----------
      10
```



Esempio - 1

Divisione decimale fra i numeri a = 7 e b = 2 su 4 bit. a : b = ?

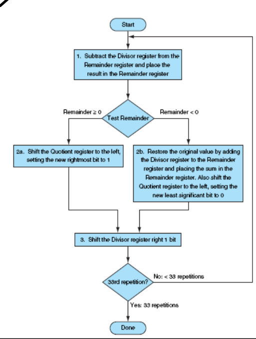
Inizializzo il divisore alla sinistra delle quattro cifre significative. La prima cifra del quoziente sarà sempre 0.

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

Divisore = $2x2^4$ = 32

Resto = 7

7 – 32 = -25 < 0



Esempio - 2

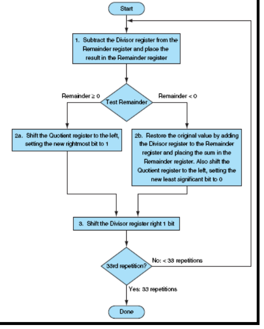
Divisione decimale fra i numeri a = 7 e b = 2   a : b = ?

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

Divisore = $2x2^3 = 16$

Resto = 7

7 – 16 = -9 < 0



Esempio - 3

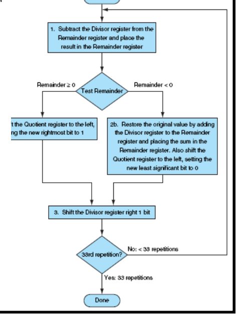
Divisione decimale fra i numeri a = 7 e b = 2   a : b = ?

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

Divisore = $2x2^2 = 8$

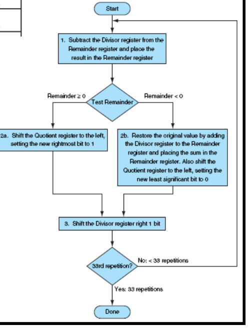
Resto = 7

7 – 8 = -1 < 0



Esempio - 4

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



Divisore = $2x2^1$ = 4

Resto = 7

7 – 4 = +3 > 0 nuovo resto parziale

Esempio - 5

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

Divisore = $2x2^0$ = 2
Resto = 3
3 – 2 = +1 > 0 nuovo resto = resto finale

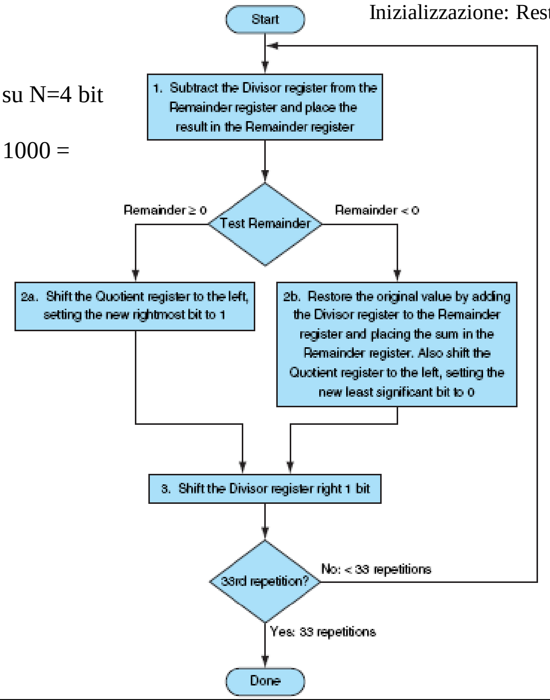
Divisione:: esempio II

Inizializzazione: Resto = Dividendo

Numeri binari su N=4 bit

0100 1010 : 1000 =

74 : 8 =



Divisione:: passo 1

01001010 : 1000 = 0

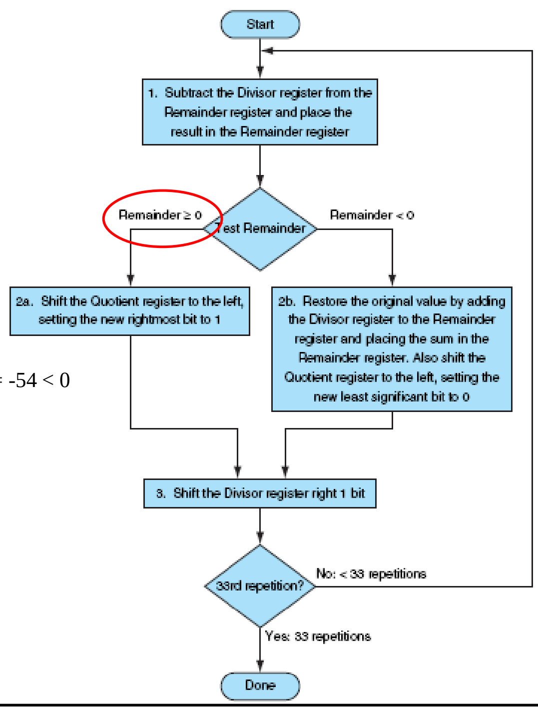
01001010 -

1000 ---- $8x2^4$

11001010 (74 - 8*16) = -54 < 0

1000 - - - -

01001010

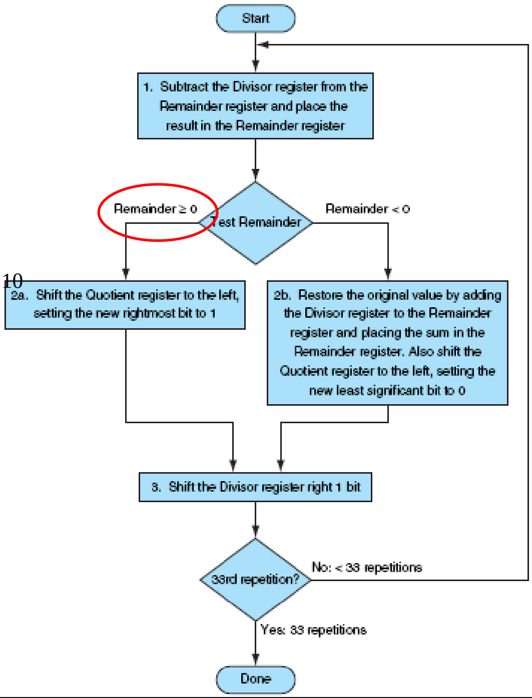


Divisione:: passo 2

01001010 : 1000 = 01

01001010-
- 1000--- $8x2^3$

00001010 > 0   (74-8*8) = 10



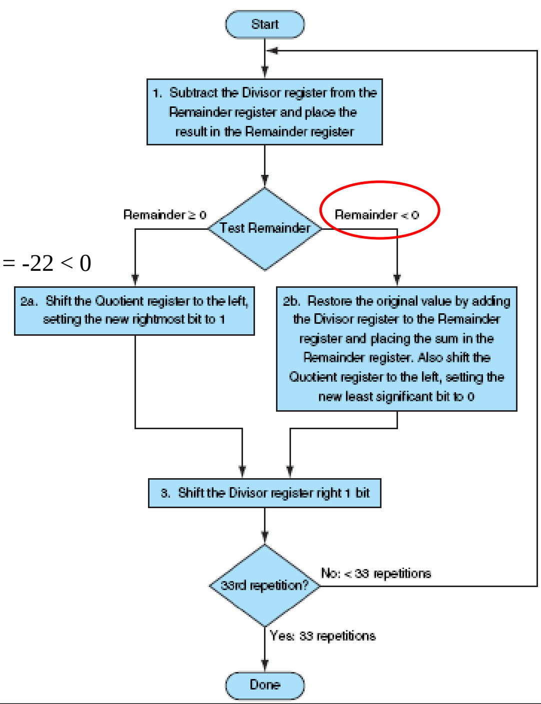
Divisione:: passo 3

01001010 : 1000 = 010

00001010 -
- -1000 -- $8x2^2$

11101011+   (10-8*4) = -22 < 0
- -1000 --

00001010



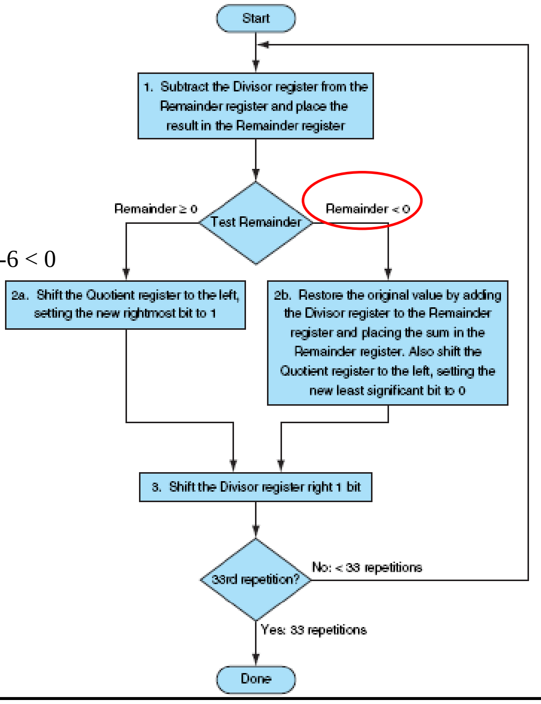
Divisione:: passo 4

01001010 : 1000 = 0100

00001010 - 10
- - -1000 - $8x2^1$

11110110 + (10-8*2) = -6 < 0
- - -1000 -

00001010 10

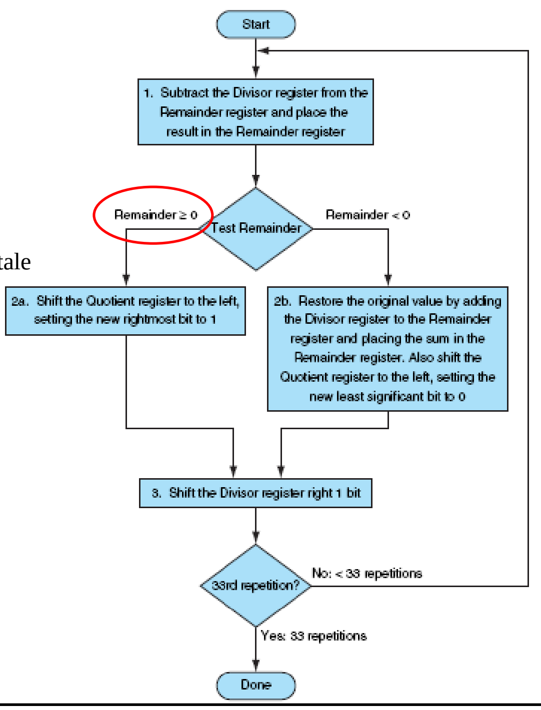


Divisione:: passo 5

01001010 : 1000 = 01001

00001010 - 10
- - - -1000 $8x2^0$

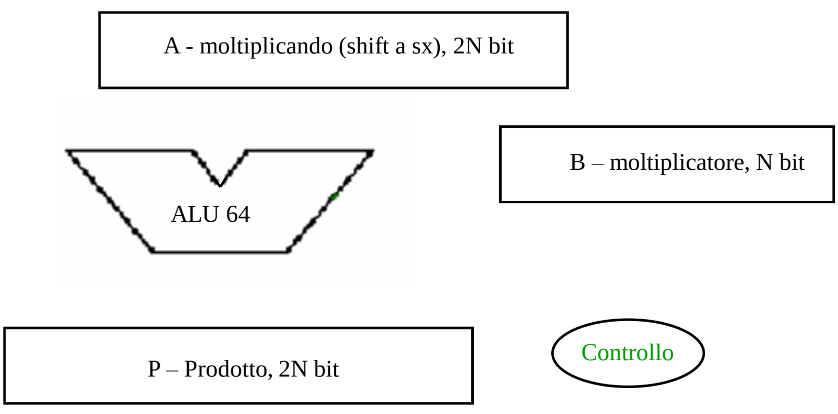
00000010 2 = resto totale



Sommario

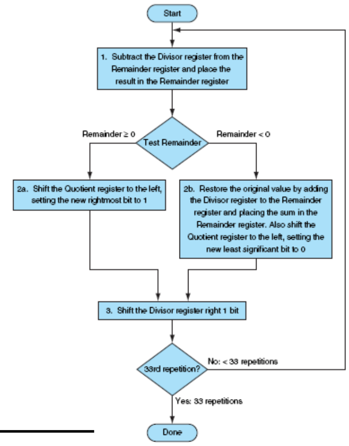
- Divisione intera
- Circuiti divisione intera
- Divisione e moltiplicazione

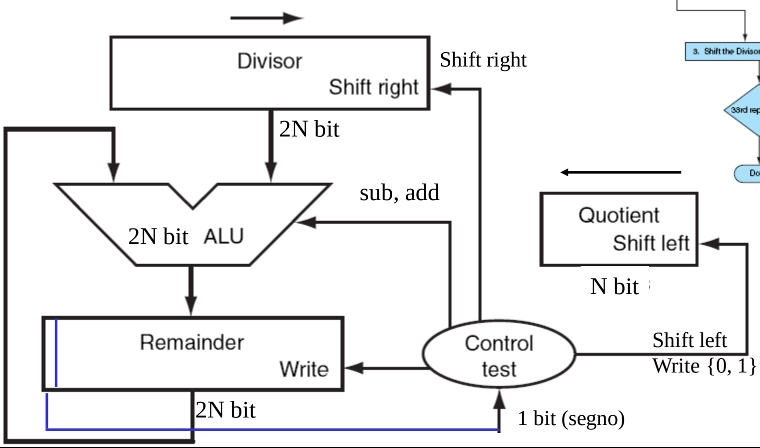
Implementazione circuitale gli stessi attori della moltiplicazione



Il circuito firmware della divisione

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0



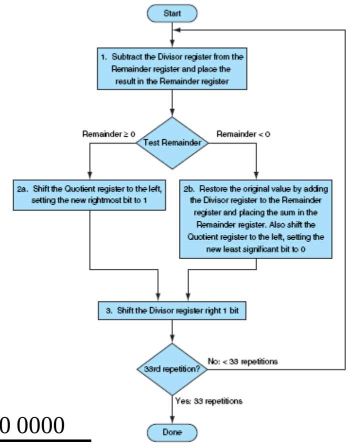


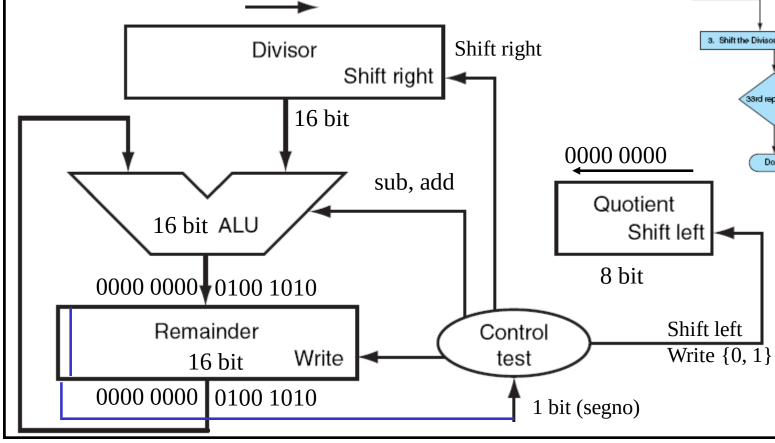
Il circuito firmware della divisione

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0
-- Parole su N = 8 bit

0100 1010 : 0000 1000 =
74 : 8 su 8 bit

0000 **1000** 0000 0000 = 2048_{10}





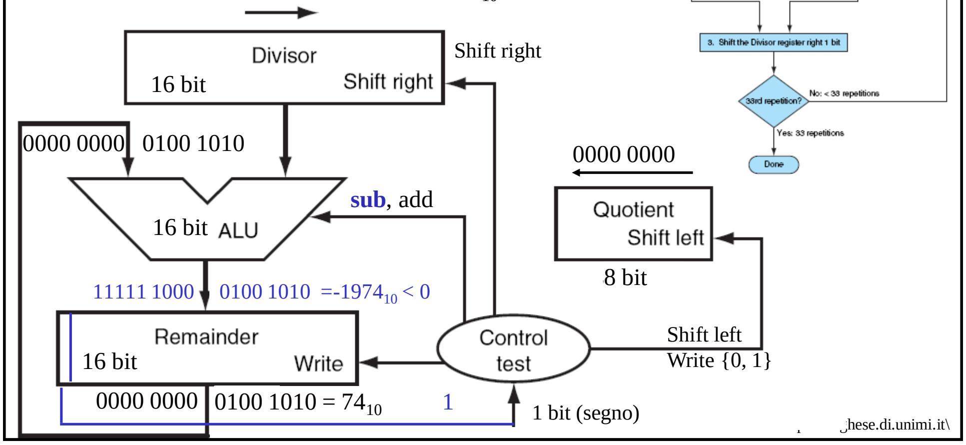
Il circuito firmware della divisione – passo 1a

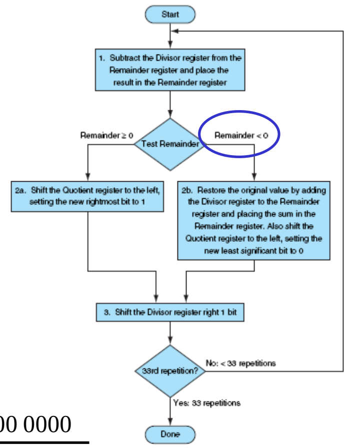
Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

74 - 8 x 2^8 = -1974 < 0

0000 **1000** 0000 0000 = 2048_{10}





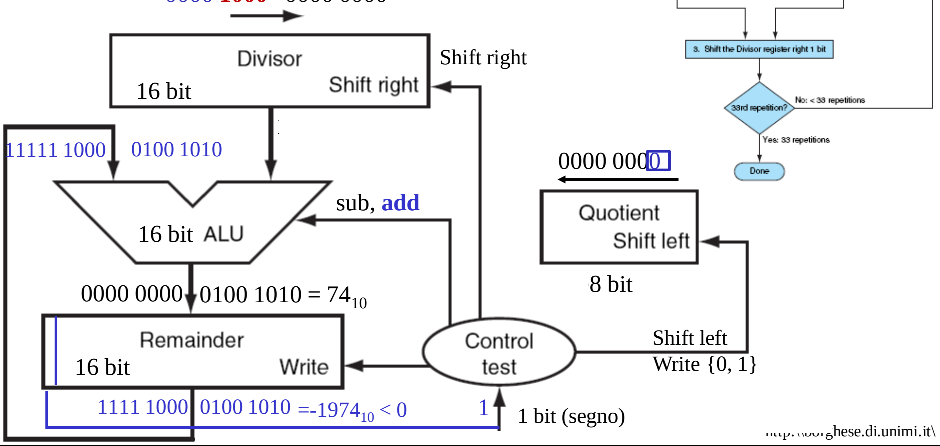
Il circuito firmware della divisione – passo 1b

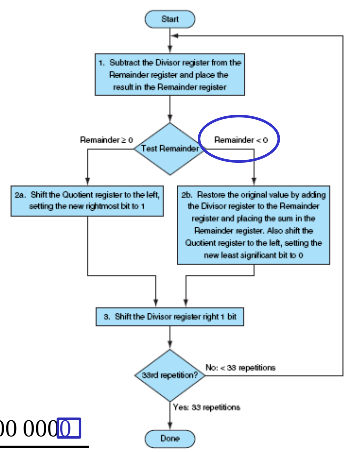
Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

-1974 + 8 x 2^8 = +74 – Q_1 = 0

0000 **1000** 0000 0000



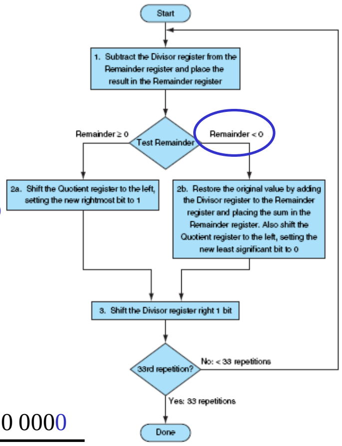


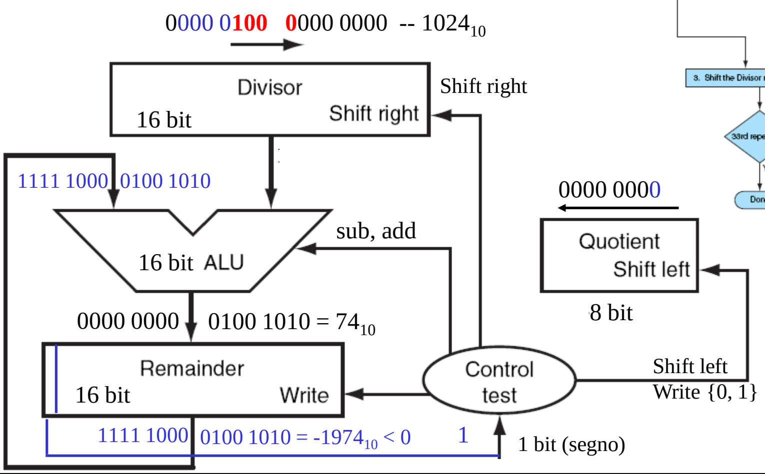
Il circuito firmware della divisione – passo 1c

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

Divisore: 2048_{10} ShiftDx 1 -> 1024_{10}



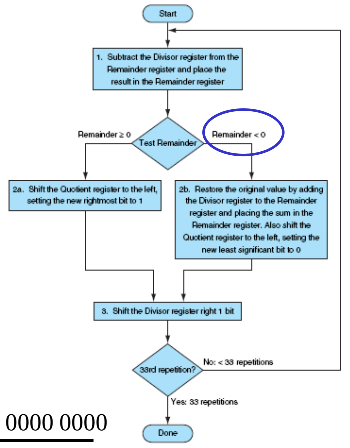


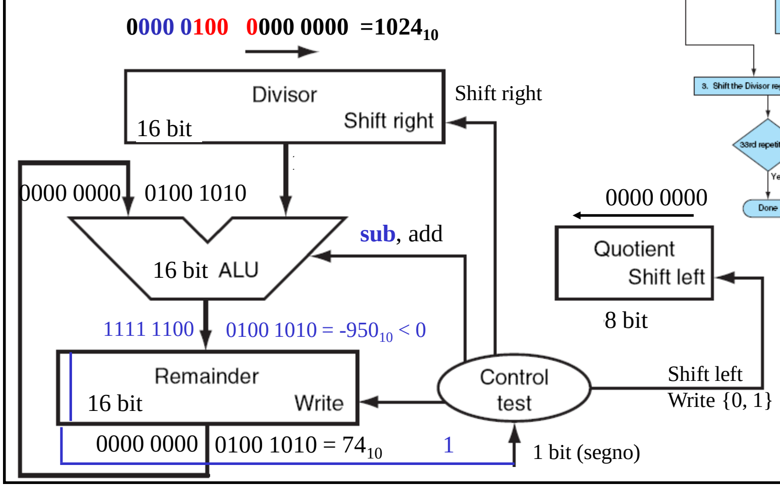
Il circuito firmware della divisione – passo 2a

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$74 - 8 \times 2^{10} = -1024 < 0$



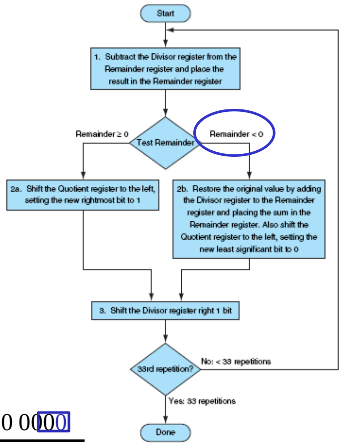


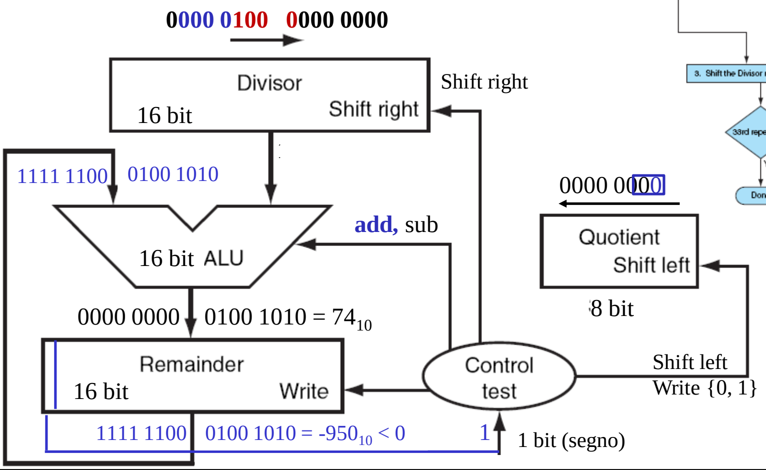
Il circuito firmware della divisione – passo 2b

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

74 - 8 x 2^7= -950 < 0 -- Q_k = 0



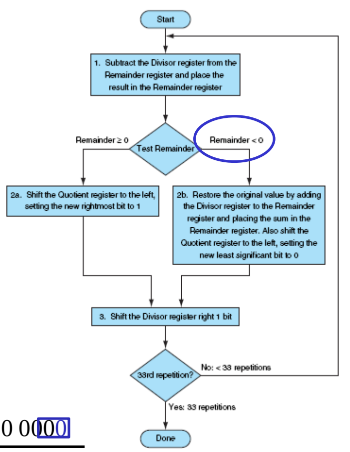


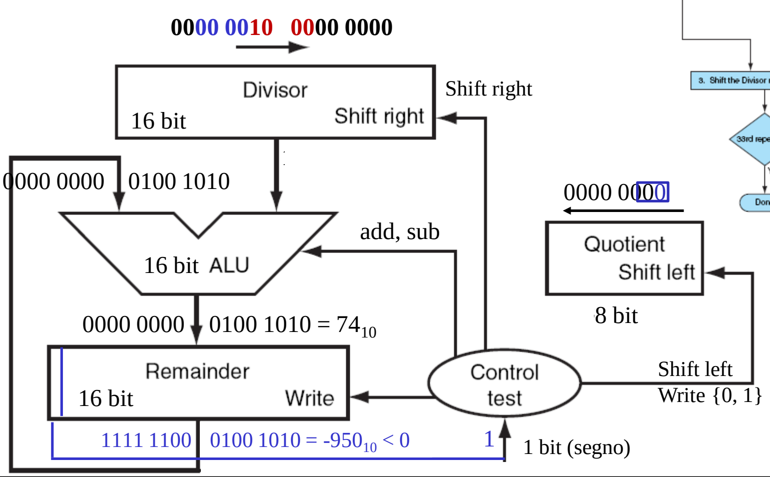
Il circuito firmware della divisione – passo 2c

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

Divisore: 1024_{10} ShiftDx 1 -> 512_{10}





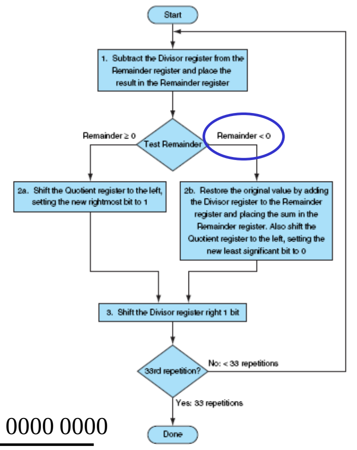
Il circuito firmware della divisione – passo 3a

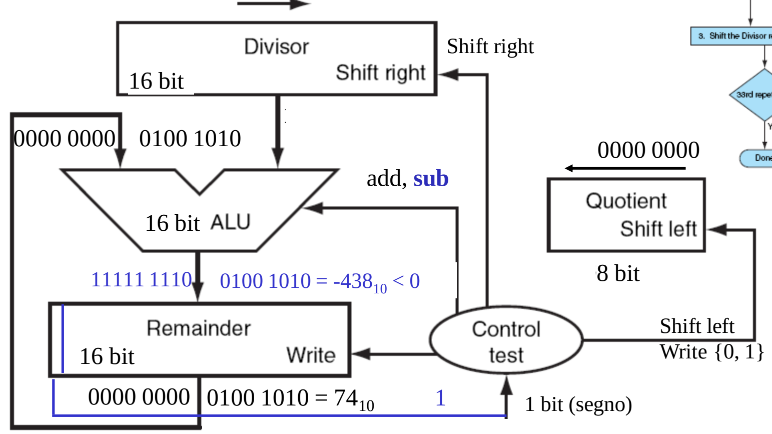
Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$74 - 8 \times 2^6 = -438 < 0$

0000 0010 0000 0000 = 512_{10}





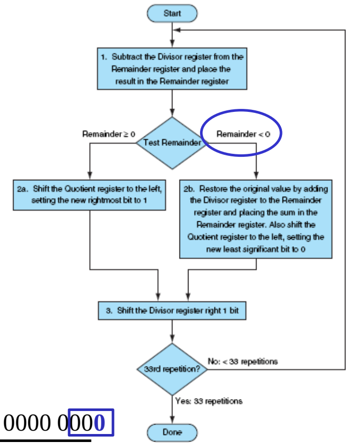
Il circuito firmware della divisione – passo 3b

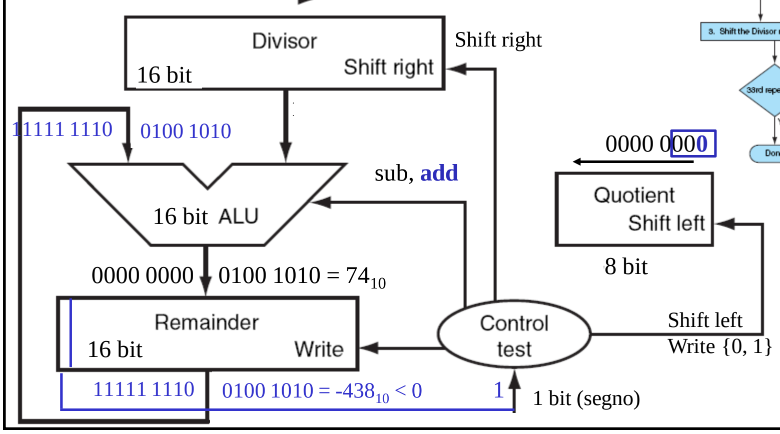
Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$74 - 8 \times 2^6 = -438 < 0$ -- $Q_k = 0$

0000 0010 0000 0000



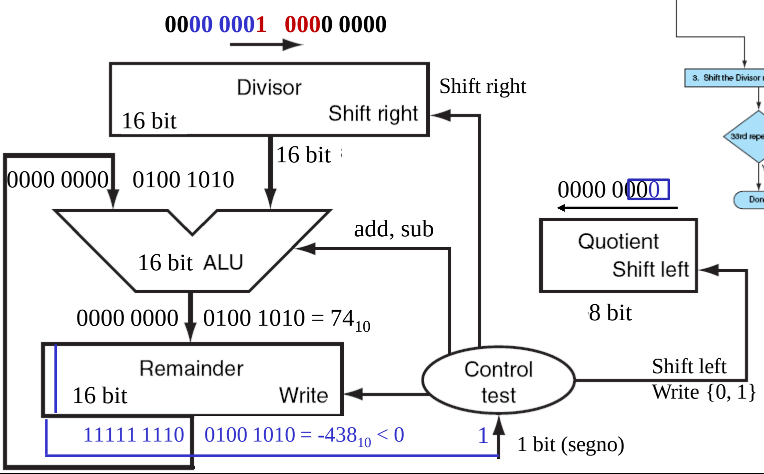


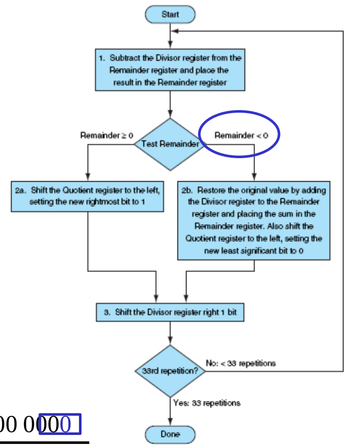
Il circuito firmware della divisione – passo 3c

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

Divisore: 512_{10} ShiftDx 1 -> 256_{10}



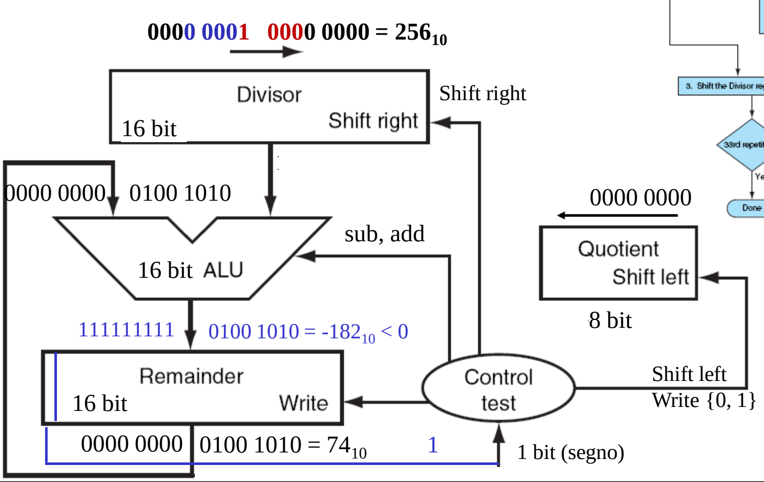


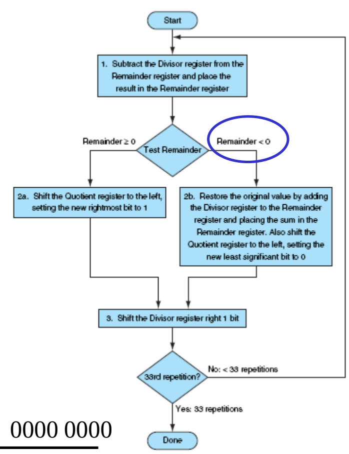
Il circuito firmware della divisione – passo 4a

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$74 - 8 \times 2^5 = -182 < 0$



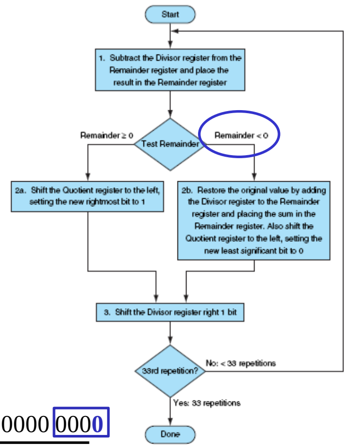


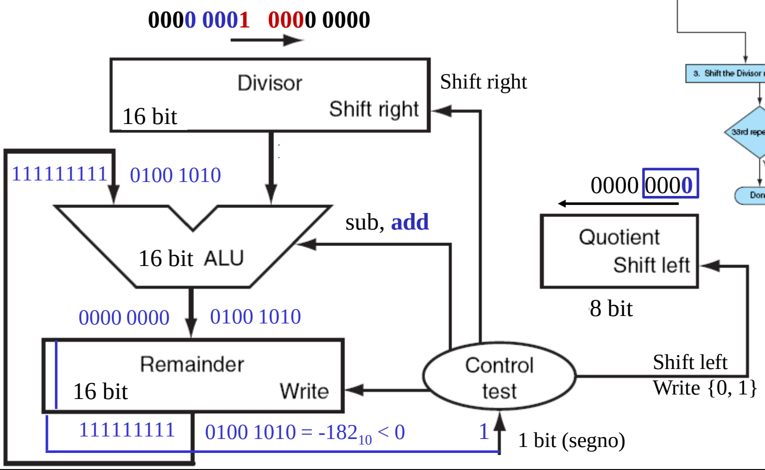
Il circuito firmware della divisione – passo 4b

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$74 - 8 \times 2^5 = -182 < 0$ -- $Q_k = 0$





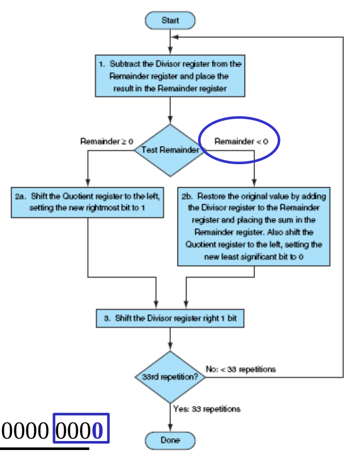
http:\\borghese.di.unimi.it\

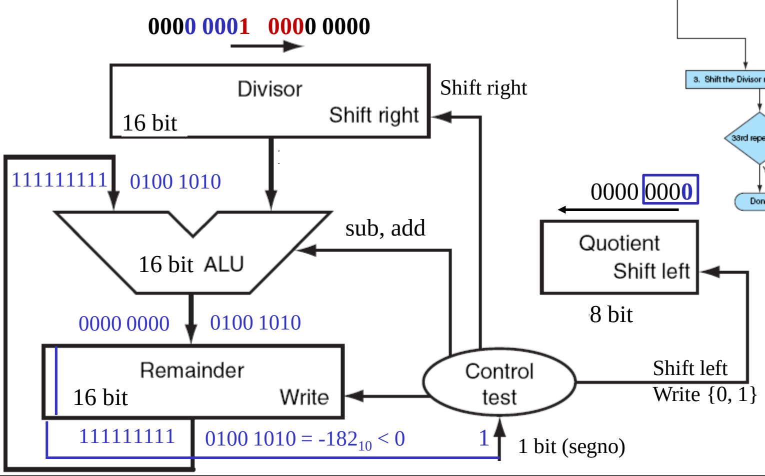
Il circuito firmware della divisione – passo 4c

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$74 - 8 \times 2^5 = -182 < 0$ -- $Q_k = 0$





http:\\borghese.di.unimi.it\

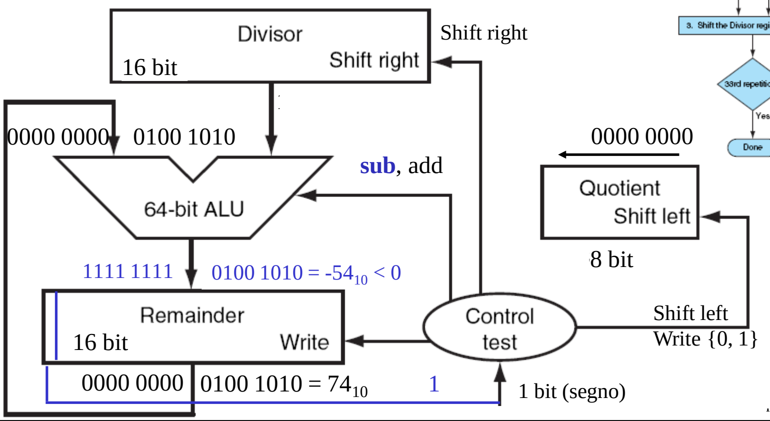
Il circuito firmware della divisione – passo 5a

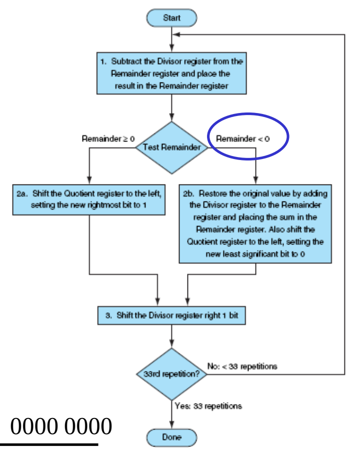
Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

74 - 8 x 2⁴ = -54 < 0

0000 **0000** **1000** 0000 = **128**$_{10}$





http:\\borghese.di.unimi.it\

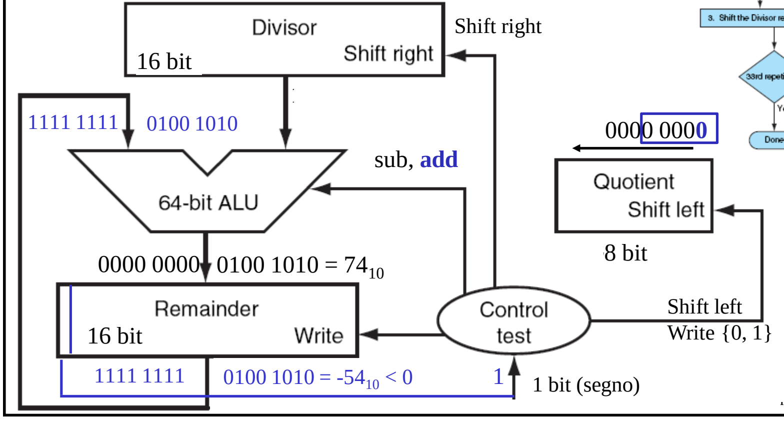
Il circuito firmware della divisione – passo 5b

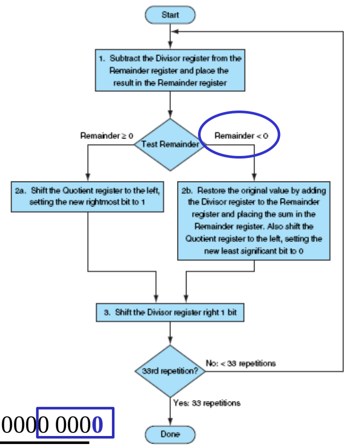
Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

74 - 8 x 2⁴ = -54 < 0

0000 **0000** **1000** 0000





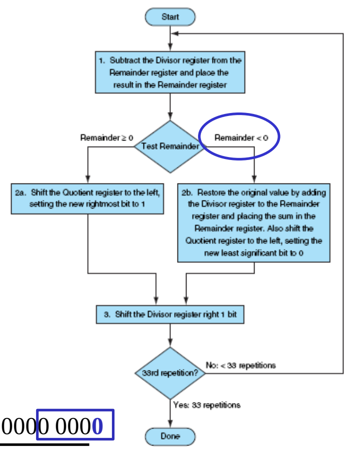
http:\\borghese.di.unimi.it\

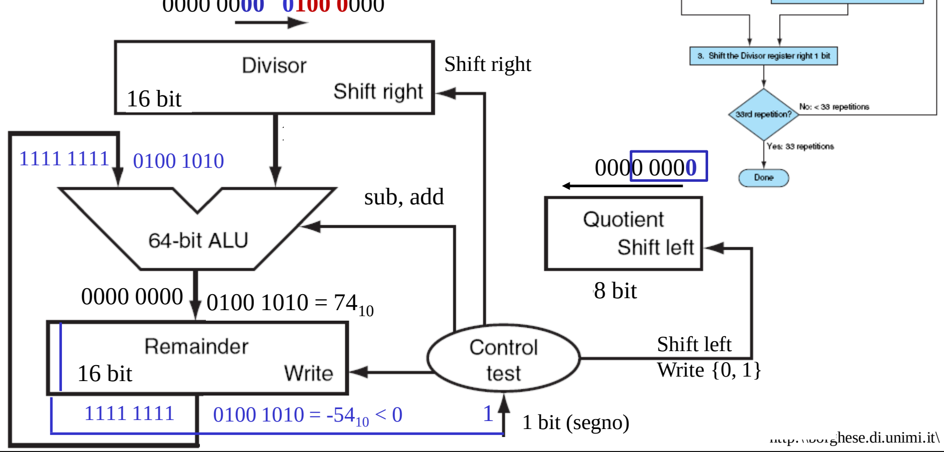
Il circuito firmware della divisione – passo 5c

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

74 - 8 x 2^4 = -54 < 0 -- Q_k = 0





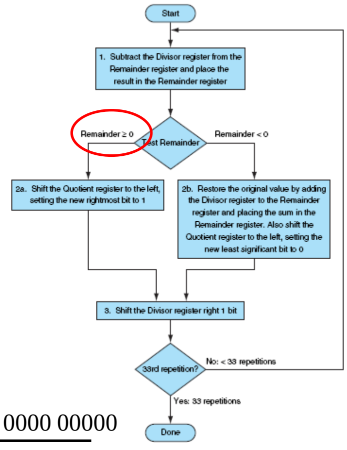
http:\\borghese.di.unimi.it\

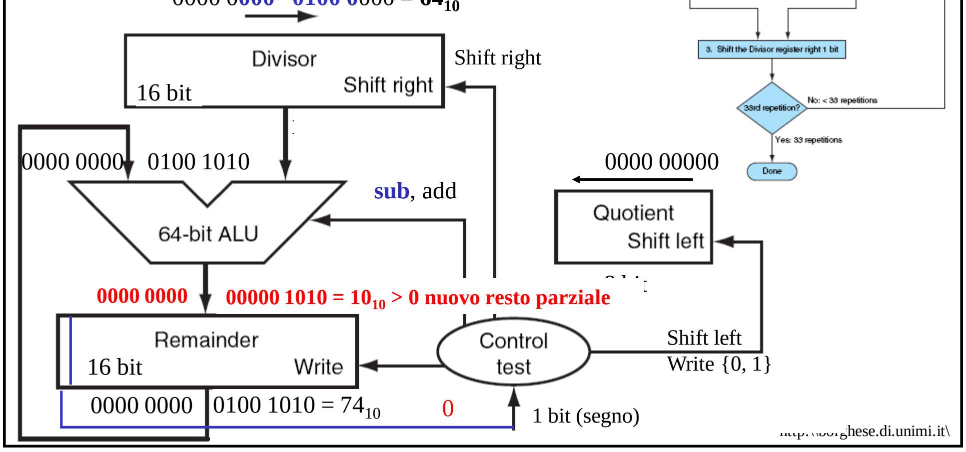
Il circuito firmware della divisione – passo 6a

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

74 - 8 x 2^3 = +10 ≥ 0





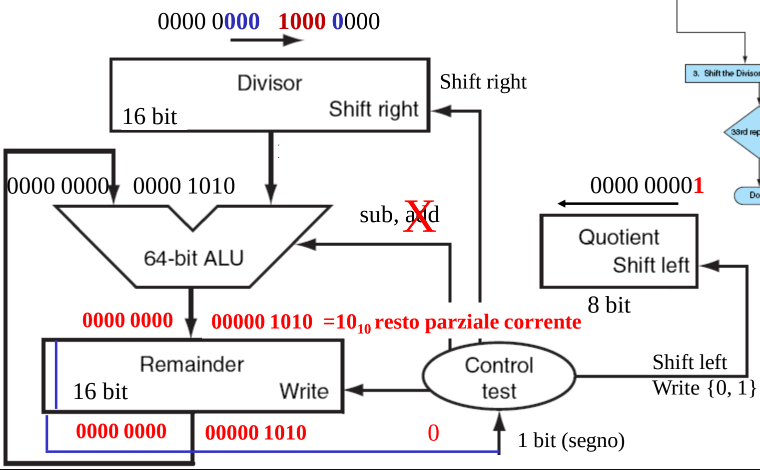
http:\\borghese.di.unimi.it\

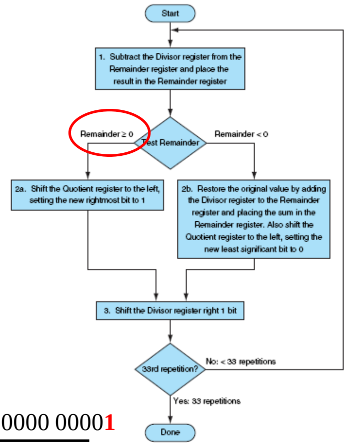
Il circuito firmware della divisione – passo 6b

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$74 - 8 \times 2^3 = +10 \geq 0$



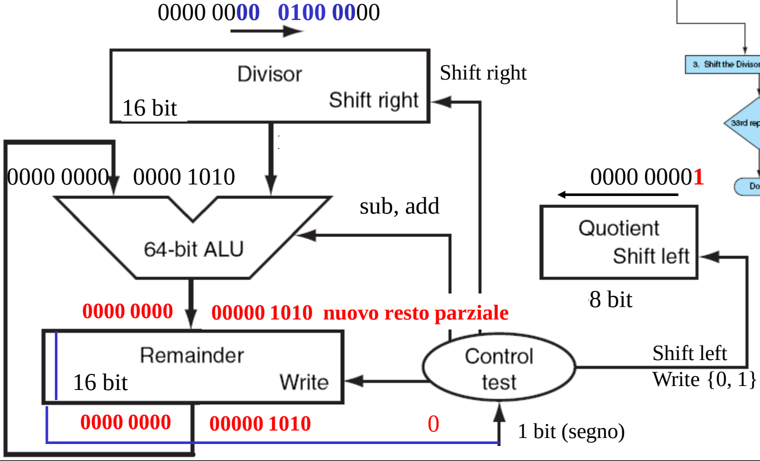


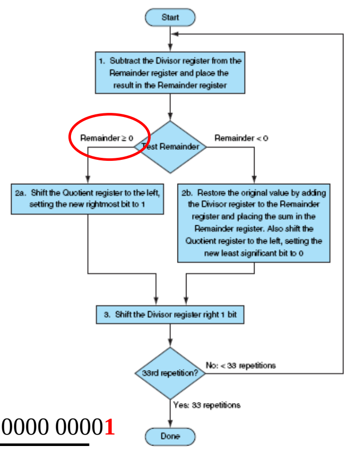
Il circuito firmware della divisione – passo 6c

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$74 - 8 \times 2^3 = +10 \geq 0$ -- $Q_k = 1$



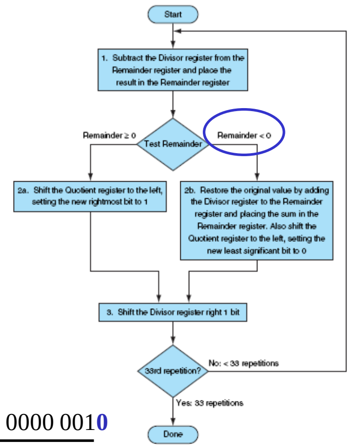


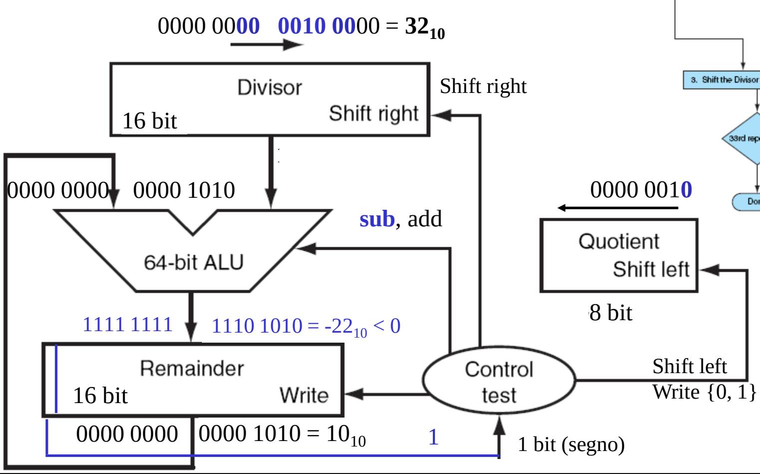
Il circuito firmware della divisione – passo 7a

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$10 - 8 \times 2^2 = -22 < 0$





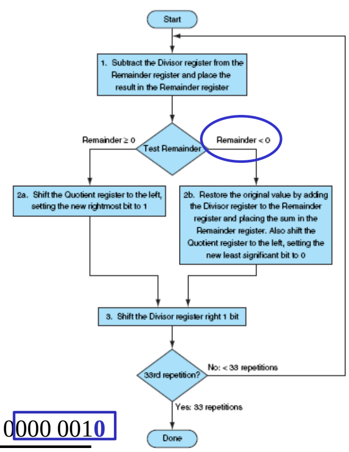
http:\\borghese.di.unimi.it\

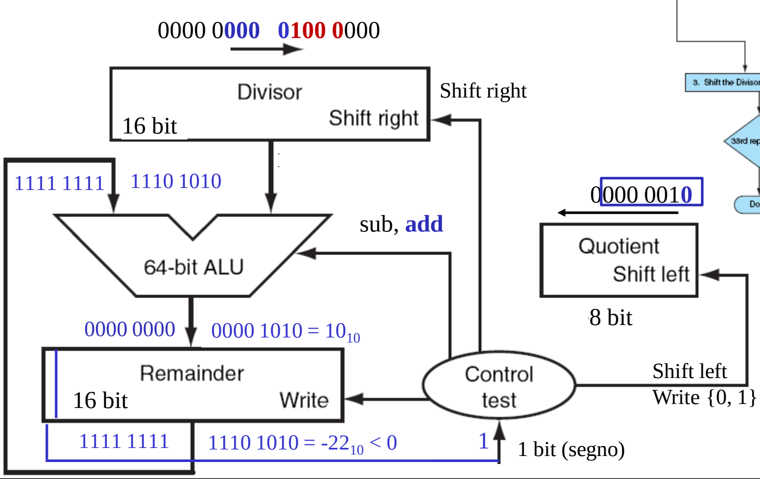
Il circuito firmware della divisione – passo 7b

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$10 - 8 \times 2^2 = -22 < 0$





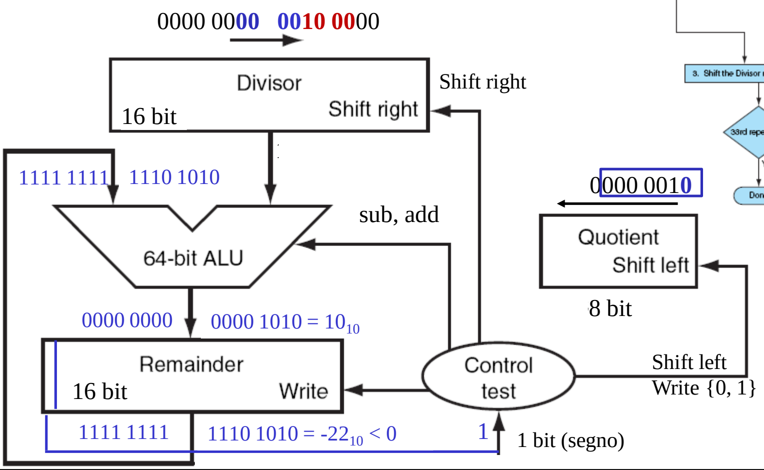
http:\\borghese.di.unimi.it\

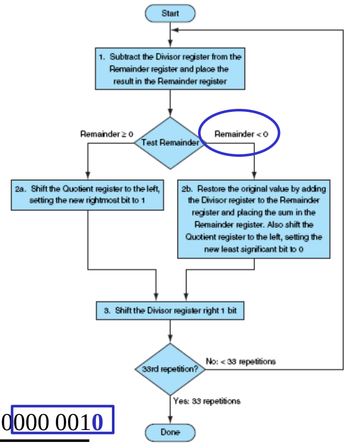
Il circuito firmware della divisione – passo 7c

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$10 - 8 \times 2^2 = -22 < 0$ -- $Q_k = 0$



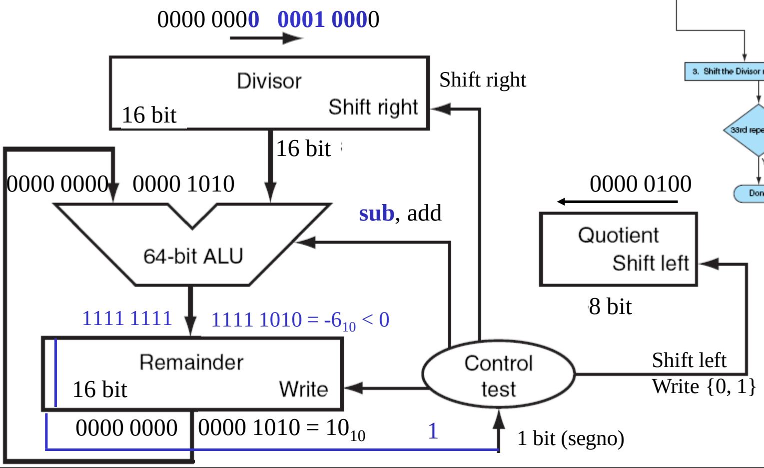


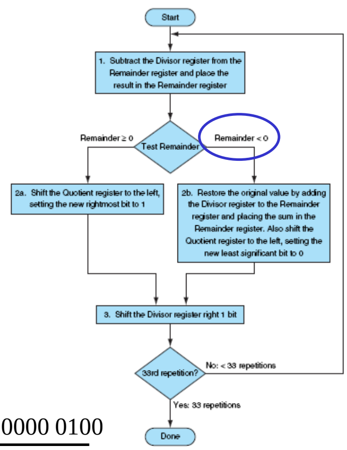
Il circuito firmware della divisione – passo 8a

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$10 - 8 \times 2^1 = -6 < 0$



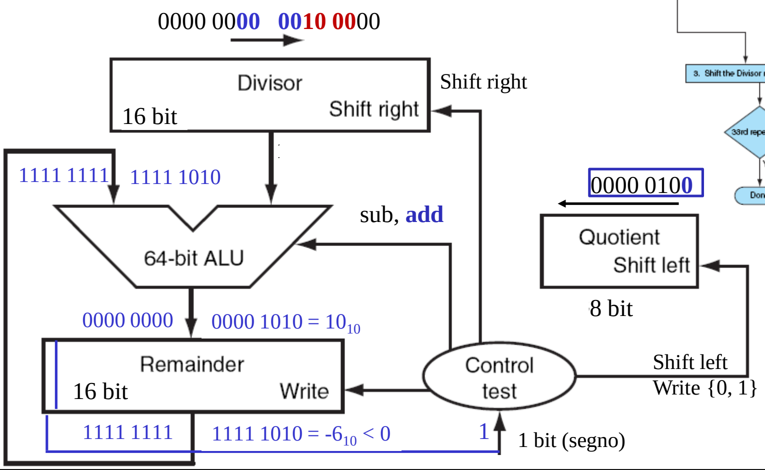


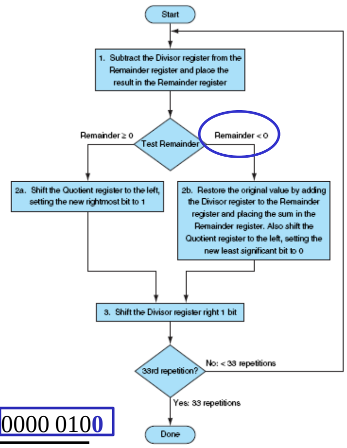
Il circuito firmware della divisione – passo 8b

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$10 - 8 \times 2^1 = -6 < 0$





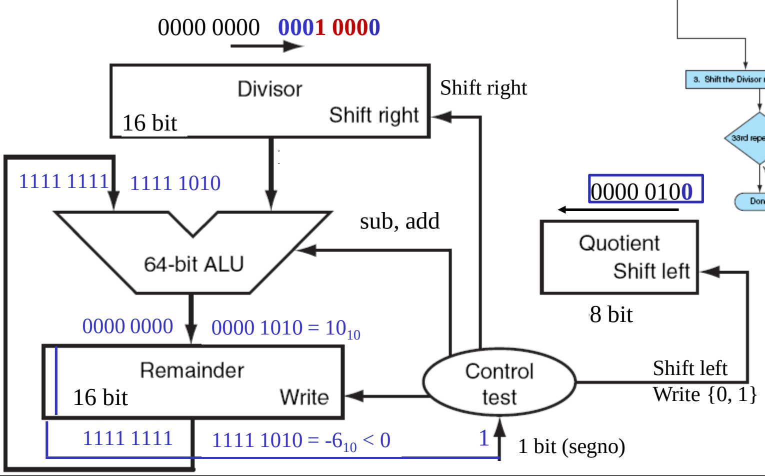
http:\\borghese.di.unimi.it\

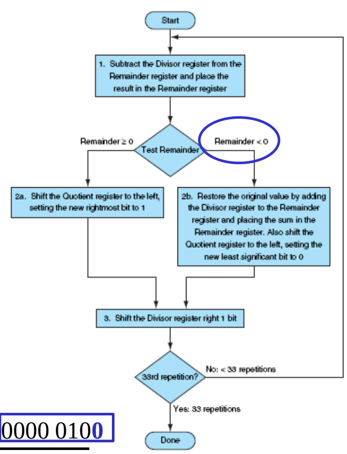
Il circuito firmware della divisione – passo 8c

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

$10 - 8 \times 2^1 = -6 < 0$ -- $Q_k = 0$





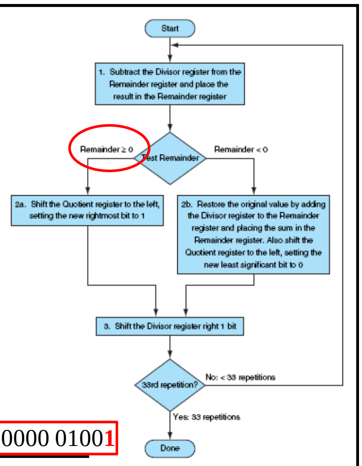
http:\\borghese.di.unimi.it\

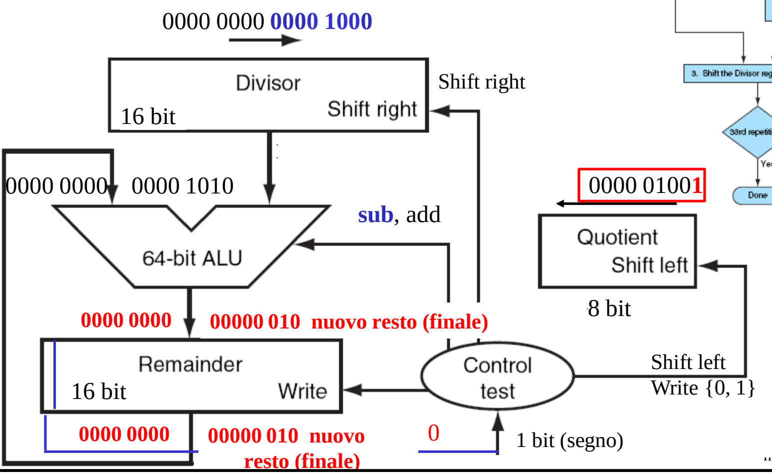
Il circuito firmware della divisione – passo 9

Inizializzazione:
-- Resto = 0 | Dividendo
-- Divisore | 0

1001010 : 1000 =

10 - 8 x 2^0 = +2 ≥ 0



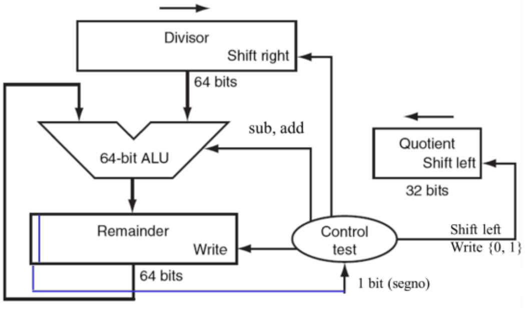


Come ottimizzare il circuito della divisione

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.

Inizializziamo il resto come RESTO = 0 | DIVIDENDO.

Ad ogni passo sposto il dividendo alias resto parziale di una posizione a sx ed inserisco un bit del quozie



Esempio - 1

Divisione decimale fra i numeri a = 7 e b = 2 su 4 bit. a : b = ?

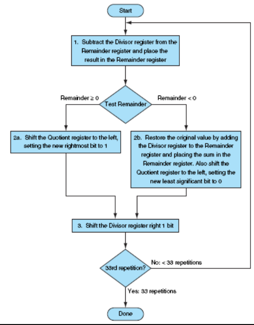
Inizializzo il divisore alla sinistra delle quattro cifre significative. La prima cifra del quoziente sarà sempre 0.

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

Divisore = $1x2^4x2^1 = 32$

Resto = 7

7 – 32 = -25 < 0



Esempio - 2

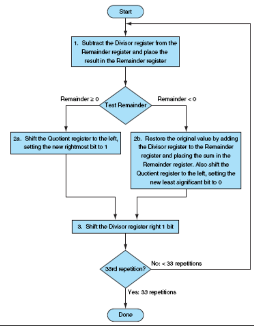
Divisione decimale fra i numeri a = 7 e b = 2 a : b = ?

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

Divisore = $1x2^3x2 = 16$

Resto = 7

7 – 16 = -9 < 0



Effetto dello spostamento relativo - I

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

Iterazione	Passo	Quoziente	Divisore	Dividendo/Resto
0	Valori iniziali	0000	0010	0 0000 0111
1	Resto = Resto - div	0000	0010	1 1110 0111
	Resto<0 ->Resto+=div;	0000	0010	0 0000 0111
	Resto << 1 - Q_0=0	0000	0010	0 0000 1110

Sposto a dx il divisore rispetto al dividendo (resto parziale)
Tengo fermo il resto parziale

⇓

Sposto a sx il dividendo (resto parziale) rispetto al divisore
Tengo fermo il divisore.

Effetto dello spostamento relativo - II

Iteration	Step	Quotient	Divisor	Remainder
		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

Iterazione	Passo	Quoziente	Divisore	Dividendo/Resto
		0000	0010	0 0000 1110
2	Resto = Resto - div	0000	0010	1 1110 1110
	Resto<0 ->Resto+=div;	0000	0010	0 0000 1110
	Resto << 1 - Q_0=0	0000	0010	0 0001 1100

Effetto dello spostamento relative - III

Iteration	Step	Quotient	Divisor	Remainder
				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

Iterazione	Passo	Quoziente	Divisore	Dividendo/Resto
		0000	0010	0 0001 1100
3	Resto = Resto - div	0000	0010	1 1111 1100
	Resto<0 ->Resto+=div;	0000	0010	0 0001 1100
	Resto << 1 - Q_0=0	0000	0010	0 0011 1000

Effetto dello spostamento relativo - IV

Iteration	Step	Quotient	Divisor	Remainder
				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

Iterazione	Passo	Quoziente	Divisore	Dividendo/Resto
		0000	0010	0 0011 1000
4	Resto = Resto - div	0000	0010	0 0001 1000
	Resto>0	0000	0010	0 0001 1000
	Resto << 1 - Q_0=1	0001	0010	0 0011 0000

Effetto dello spostamento relativo - V

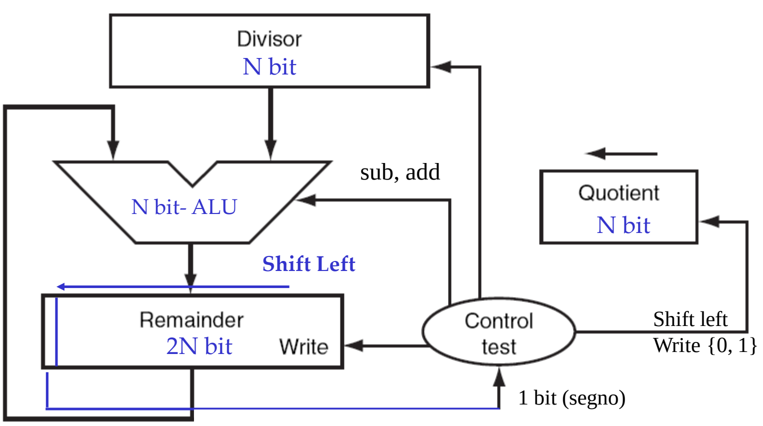
Iteration	Step	Quotient	Divisor	Remainder
				0000 0011
5	1: Rem = Rem – Div	0001	0000 0010	0000 0001
	2a: Rem ≥ 0 ⟹ sll Q, Q0 = 1	0011	0000 0010	0000 0001
	3: Shift Div right	0011	0000 0001	0000 0001

Iterazione	Passo	Quoziente	Divisore	Dividendo/Resto
		0001	0010	0 0011 0000
5	Resto = Resto - div	0001	0010	0 0001 0000
	Resto>0	0000	0010	0 0001 0000
	Resto << 1 - Q_0=1	0011	0010	0 0010 0000

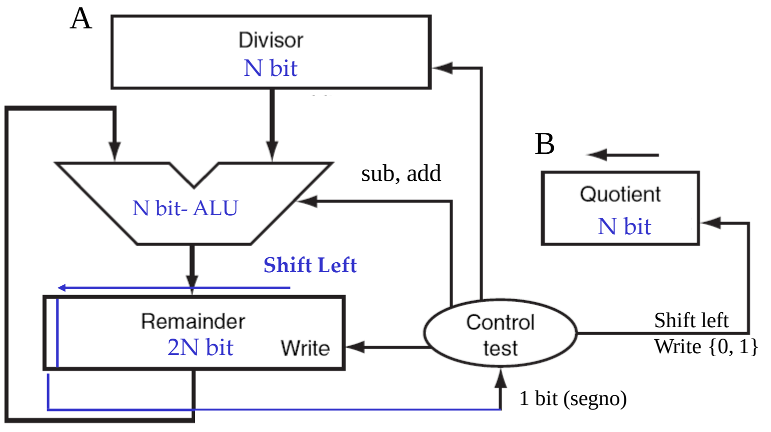
Il resto finale va fatto scorrere a dx di 1 posizione

Il circuito firmware con un'ottimizzazione

Inizializzazione: Resto = 0 | Dividendo



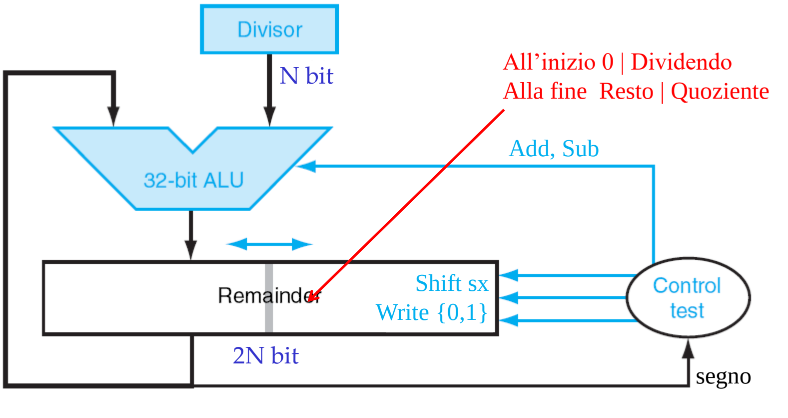
Razionale per un’ulteriore ottimizzazione



Il quoziente viene riempito un bit alla volta da dx a sx
Il dividendo viene spostato da dx a sx di una posizione

Il circuito firmware ottimizzato della divisione

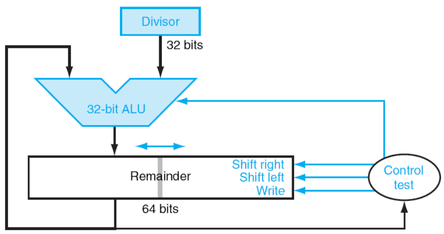
Inizializzazione: Resto = 0 | Dividendo



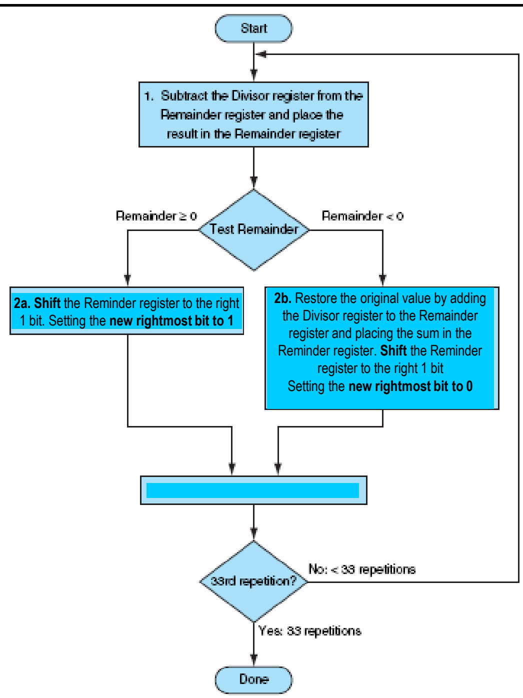
Effetto sul circuito

Iteration	Step	Quotient	Divisor	Remainder
		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

Iterazione	Passo	Divisore	Dividendo/Resto/Quoziente
		0010	0 0000 1110
2	Resto = Resto - div	0010	1 1110 1110
	Resto<0 ->Resto+=div;	0010	0 0000 1110
	Resto << 1 - Q_0=0	0010	0 0001 1100



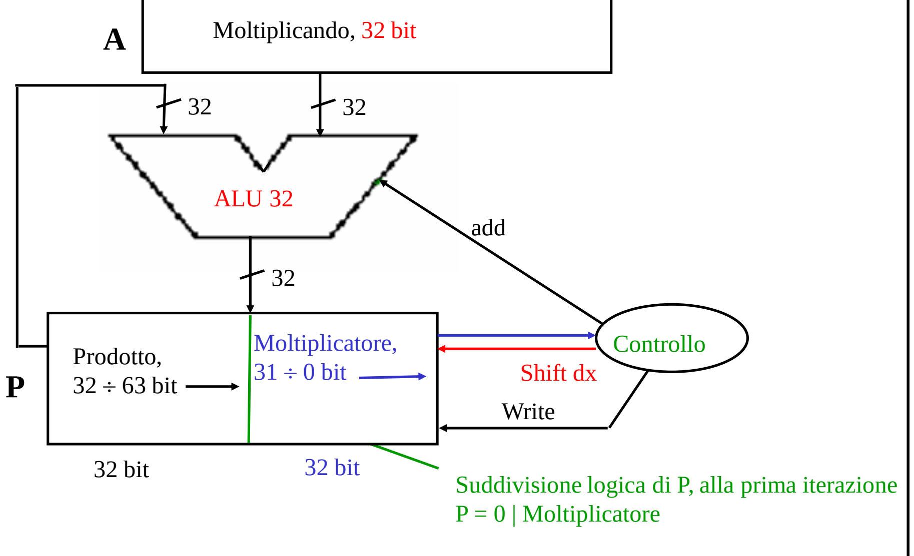
Divisione:: algoritmo



Sommario

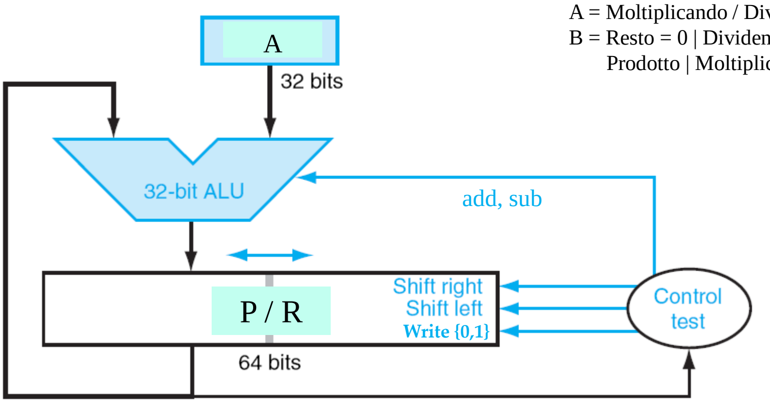
- Divisione intera
- Circuiti divisione intera
- Divisione e moltiplicazione

Circuito ottimizzato della moltiplicazione (su 32 bit)



Un unico circuito per moltiplicazione e divisione

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



Segno divisione e moltiplicazione

Moltiplicazione e divisione di numeri positivi -> viene aggiunto il segno alla fine.

Moltiplicazione: XOR dei bit di segno di moltiplicando e moltiplicatore
Se XOR positivo -> prodotto negativo

Divisione: a : b = q + r
XOR dei bit di segno di dividendo e divisore. Se XOR = 1 -> quoziente negativo.

E il resto?

$b*q = a - r$

- $a > 0$ $b > 0$ ➔ $q > 0$, $r \geq 0$ 7 : (+3) = +2 +1
- $a > 0$ $b < 0$ ➔ $q < 0$, $r \geq 0$ 7 : (-3) = -2 +1
- $a < 0$ $b > 0$ ➔ $q < 0$, $r \leq 0$ (-7) : (+3) = -2 -1
- $a < 0$ $b < 0$ ➔ $q > 0$, $r \leq 0$ (-7) : (-3) = +2 -1

Il dividendo e il resto hanno sempre lo stesso segno

Sommario

- Divisione intera
- Circuiti divisione intera
- Divisione e moltiplicazione