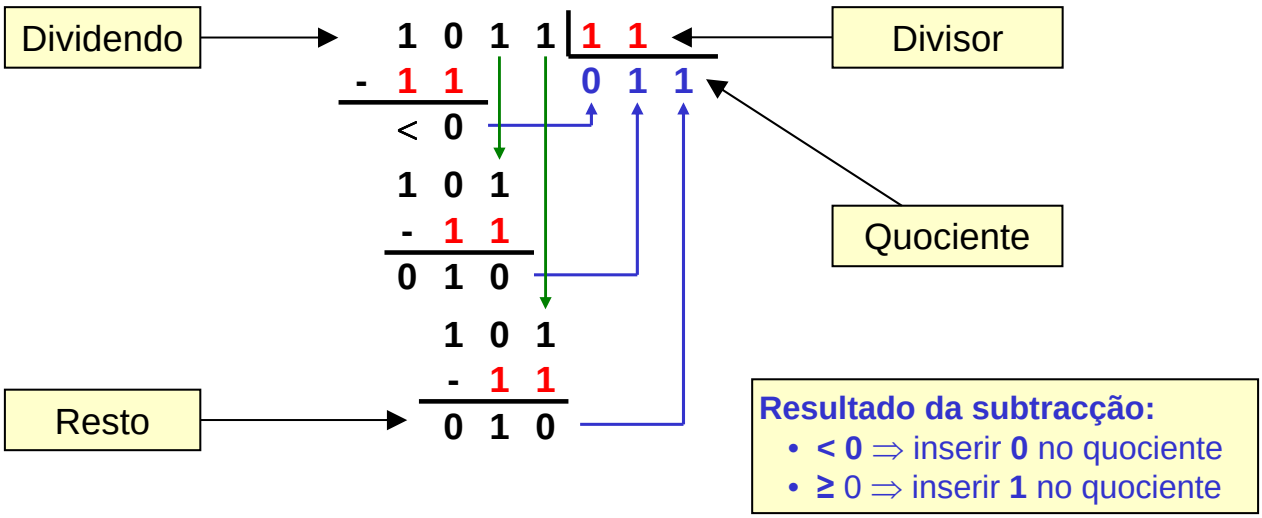
Aula 12

- Arquitectura de um divisor de números inteiros
  - Abordagem em três etapas
  - Exemplos de concretização
- Divisão de inteiros com sinal
- Divisão de inteiros no MIPS

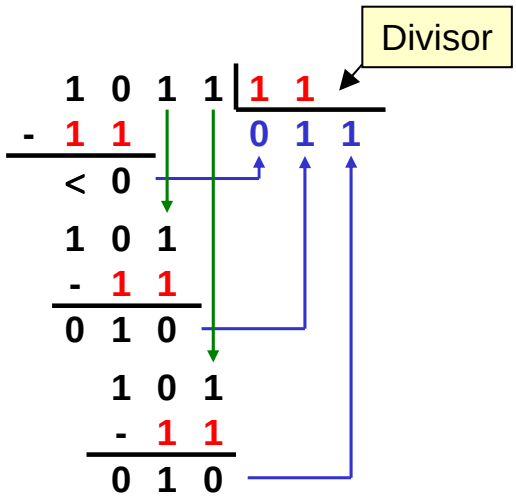
Bernardo Cunha, José Luís Azevedo, Arnaldo Oliveira, Tomás Oliveira e Silva

Divisão de inteiros em binário

Tal como acontecia com a multiplicação, também na divisão se usa uma arquitectura que aproveita o algoritmo que se ensina(va) nos primeiros anos do ensino básico. Tomemos como exemplo **1011 ÷ 0011**



Divisão de inteiros em binário



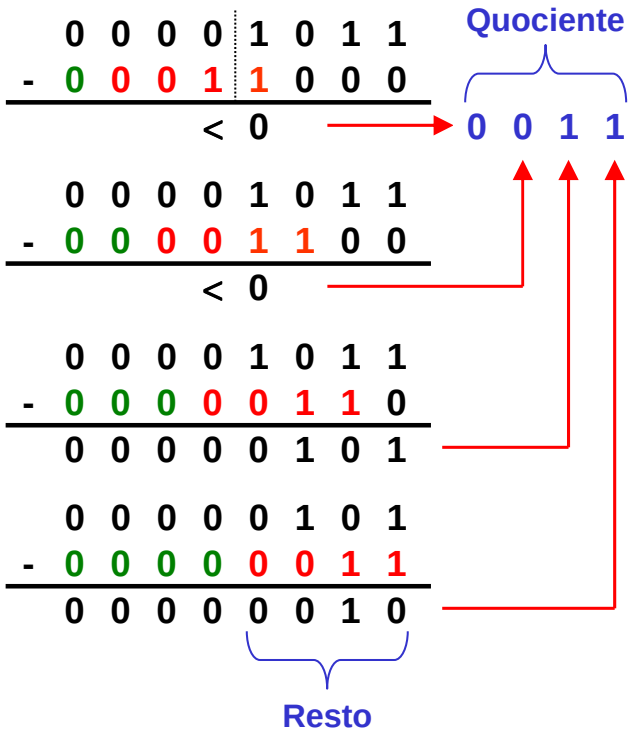
1. Começa-se por alinhar o Divisor à esquerda com o Dividendo
2. **Subtrai-se o Divisor do Dividendo**
   - Se o resultado for **positivo** (i.e. Dividendo >= Divisor) acrescenta-se "**1**" no Quociente
   - Se o resultado for **negativo** (i.e. Dividendo < Divisor) acrescenta-se "**0**" no Quociente e repõe-se o dividendo
3. Se o Divisor ainda não está alinhado à direita com o Dividendo, então desloca-se o Divisor 1 bit para a direita
4. Repete-se desde 2

- Como fazer o alinhamento do divisor à esquerda de forma automática?
- Quantas iterações são necessárias, no caso geral, para fazer a divisão?

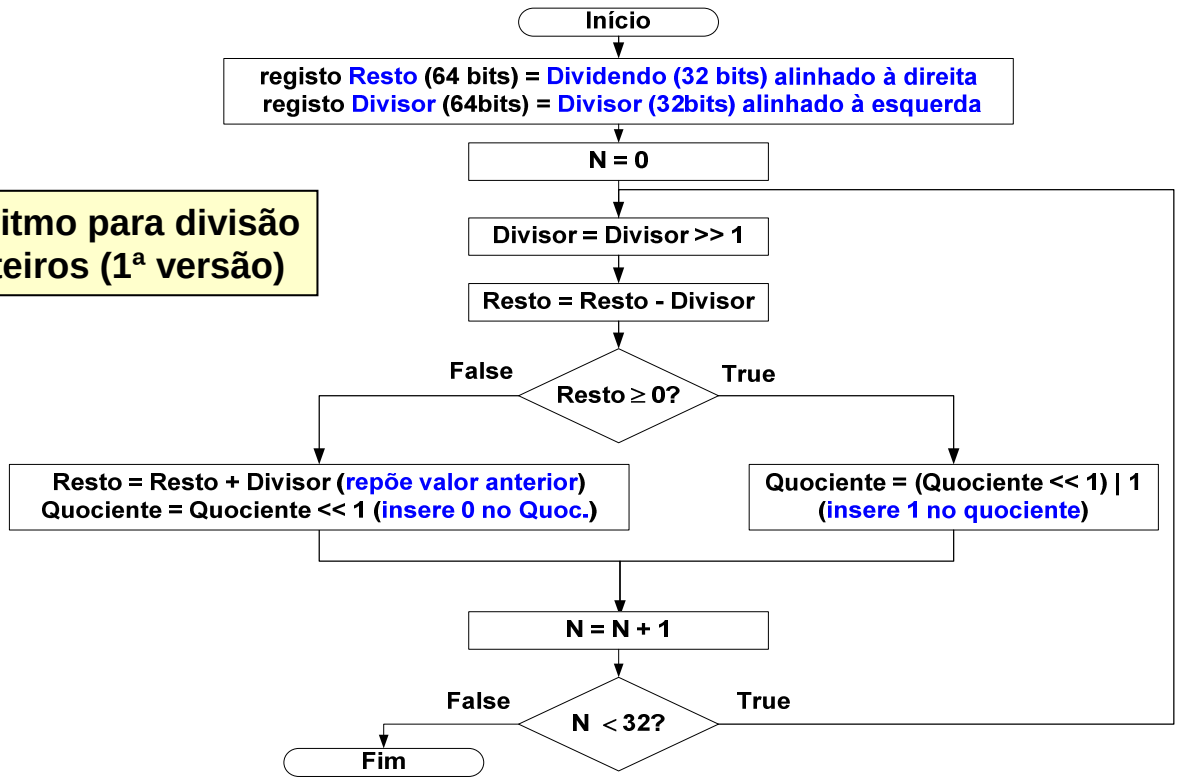
1 0 1 1 | 0 0 1 1

0 0 0 0 | 1 0 1 1 **Dividendo**
0 0 1 1 | 0 0 0 0 **Divisor**

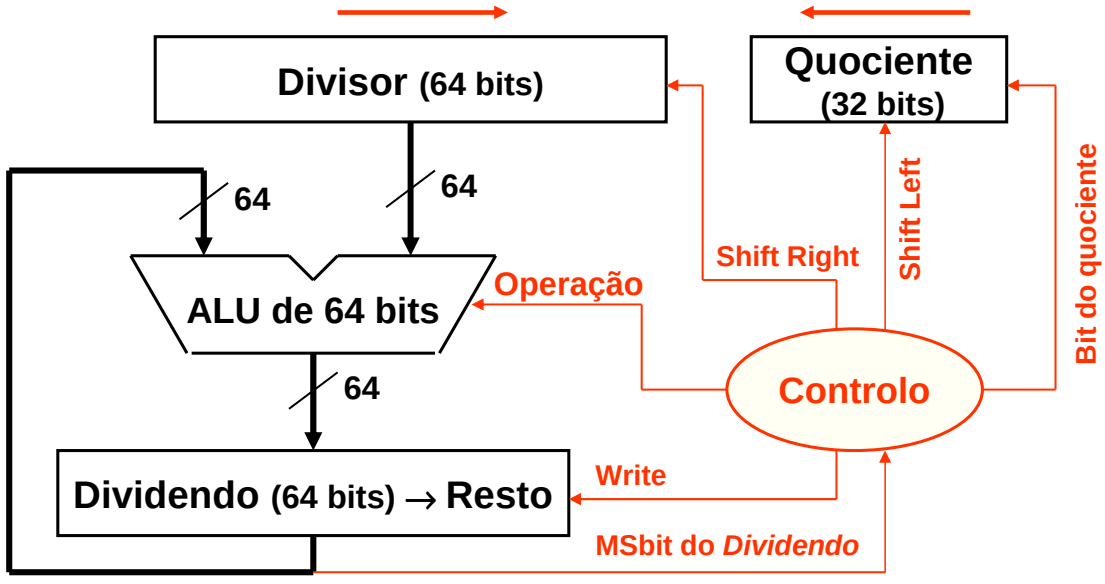
- Com operandos de 4 bits os registos para alojar o **dividendo e o divisor têm 8 bits**
- O **dividendo** é alinhado à **direita** (os 4 MSbits são colocados a 0)
- O **divisor** é alinhado à **esquerda** (os 4 LSbits são colocados a 0)
- Por cada nova iteração o **divisor é deslocado à direita 1 bit**
- O **número total de iterações** é igual ao **número de bits do dividendo original**



Algoritmo para divisão de inteiros (1ª versão)



Arquitectura de um Divisor (1ª versão)

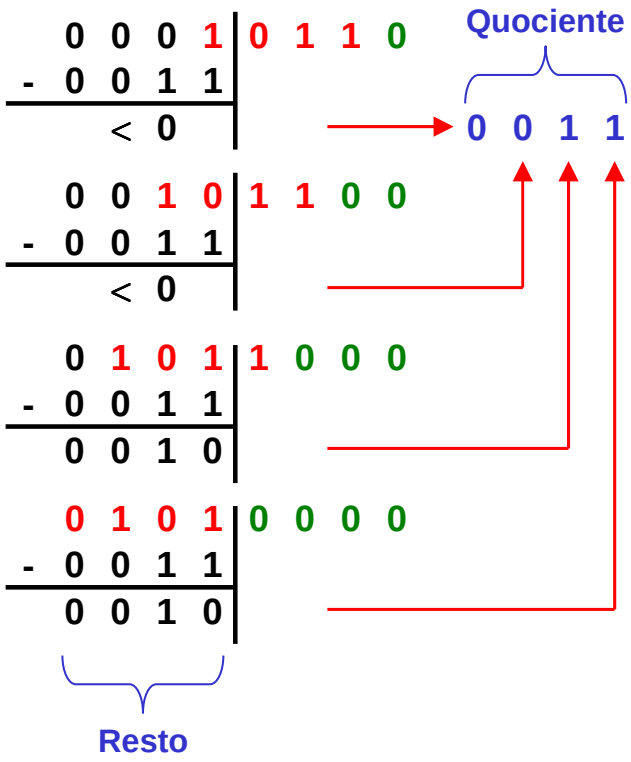


Nesta 1ª versão, a ***ALU*** e os registos ***Divisor*** e ***Dividendo/Resto*** operam com 64 bits.

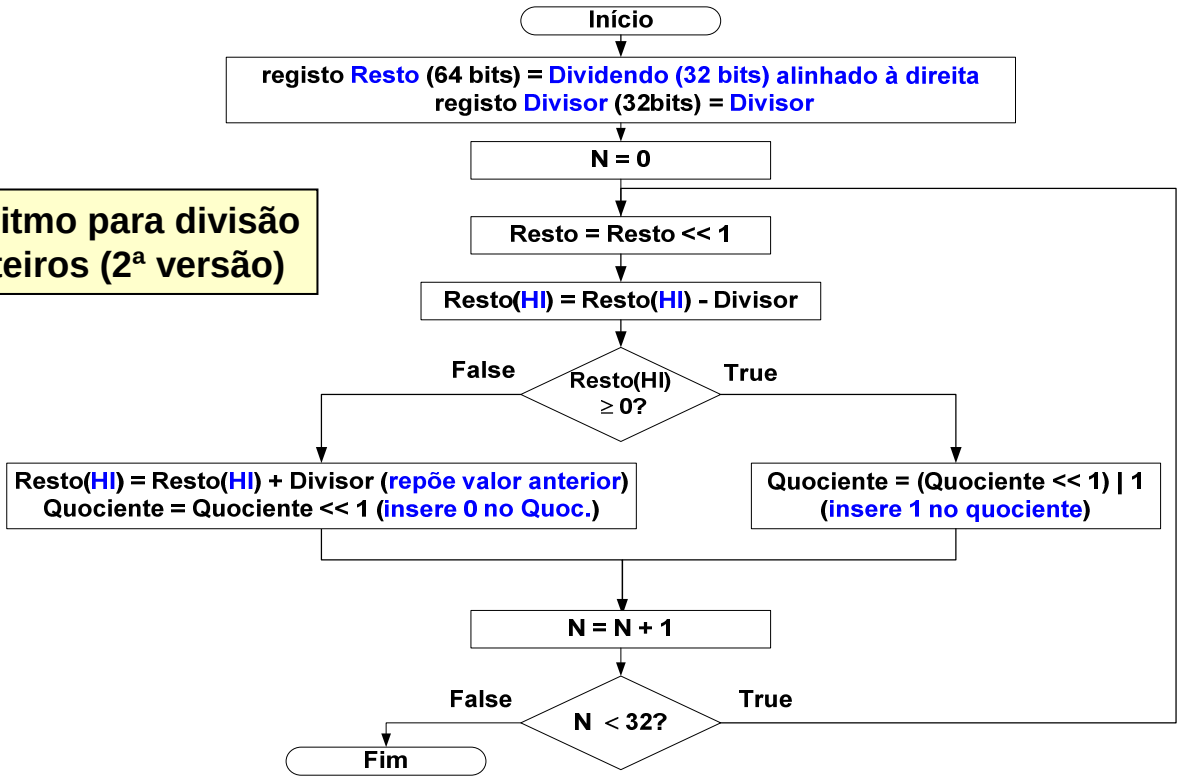
1 0 1 1 | 0 0 1 1

0 0 0 0 | 1 0 1 1 — Dividendo
0 0 1 1 — Divisor

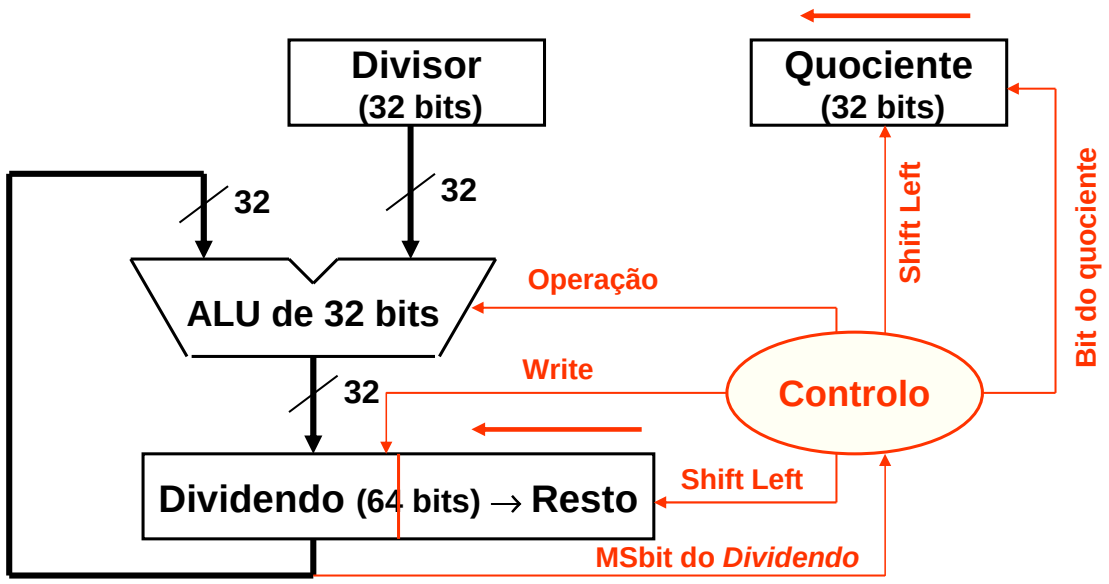
- O movimento relativo do ***Dividendo/Resto*** e do ***Divisor*** mantém-se fixando o *Divisor* e **deslocando** para a **esquerda** o ***Dividendo/Resto***
- O registo *Divisor* mantém assim a dimensão original (4 bits no exemplo)
- A **subtracção** (entre dividendo e divisor) pode também ser feita apenas com **4 bits**, reduzindo-se para metade a dimensão da ALU.



Algoritmo para divisão de inteiros (2ª versão)



Arquitectura de um Divisor (2ª versão)



Nesta 2ª versão do divisor, a ***ALU*** e o registo ***Divisor*** operam com 32 bits.

1 0 1 1 | 0 0 1 1

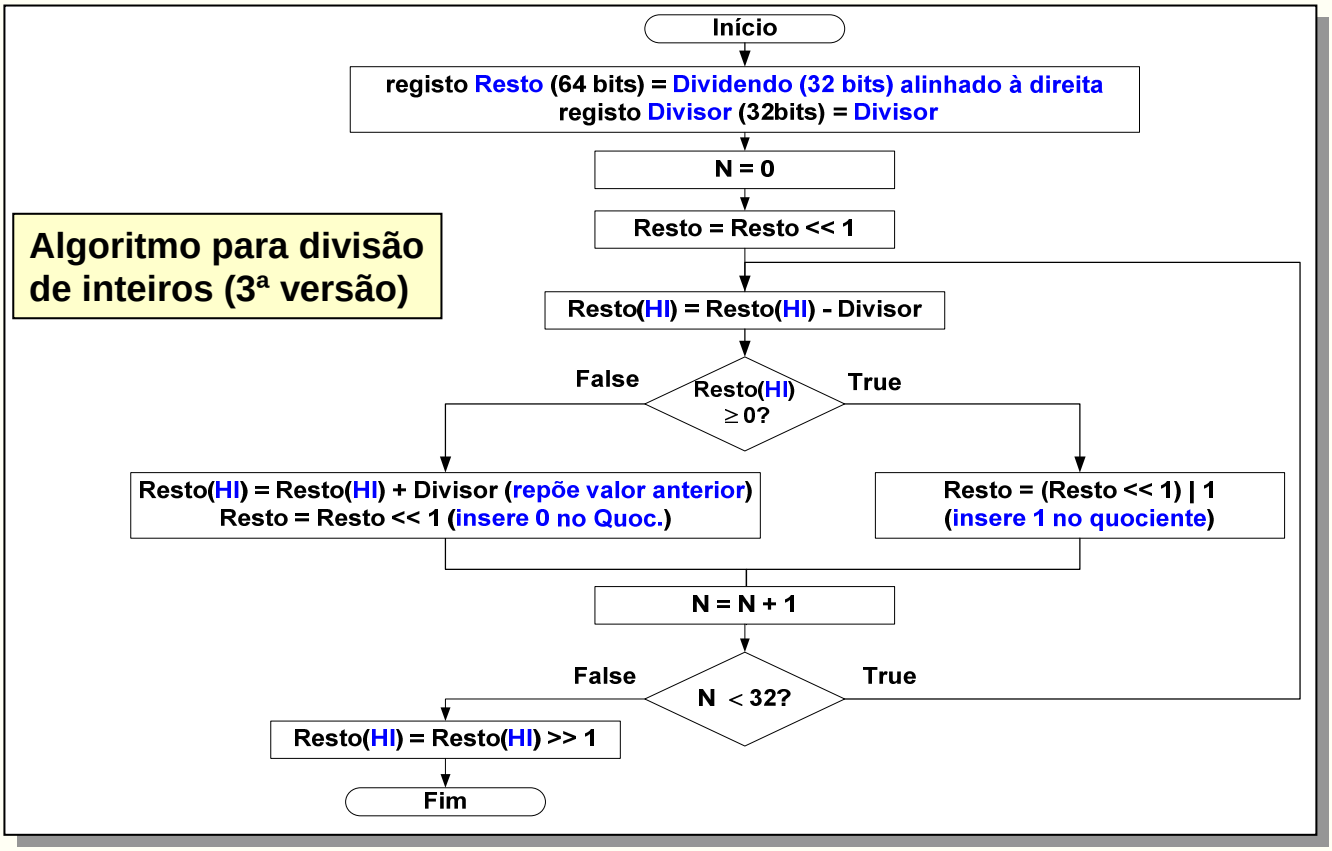
0 0 0 0 1 0 1 1 Dividendo

0 0 0 1 | 0 1 1 0 Dividendo após << 1
0 0 1 1 Divisor

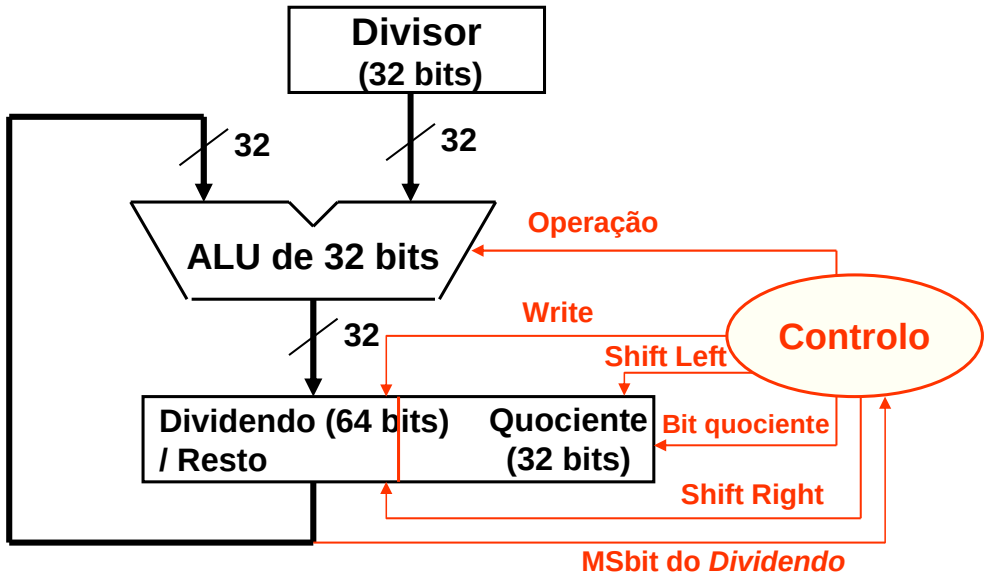
- Pode verificar-se que o deslocamento à esquerda do conteúdo do *Dividendo*, é acompanhado por um deslocamento idêntico do *Quociente*
- Em cada deslocamento à esquerda do *Dividendo* é introduzido um zero (não útil) no bit menos significativo
- **Esse espaço pode ser aproveitado para guardar o próximo bit do quociente**
- Desta forma poupa-se ainda o espaço que seria necessário para armazenar esse quociente

```
  0 0 0 1 | 0 1 1 0
- 0 0 1 1
    < 0
  0 0 1 0 1 1 0 0

  0 0 1 0 | 1 1 0 0
- 0 0 1 1
    < 0
  0 1 0 1 1 0 0 0

  0 1 0 1 | 1 0 0 0
- 0 0 1 1
  0 0 1 0
  0 1 0 1 0 0 0 1

  0 1 0 1 | 0 0 0 1
- 0 0 1 1
  0 0 1 0
  0 1 0 0 0 0 1 1

  0 0 1 0 | 0 0 1 1
    Resto   Quociente
```

Algoritmo para divisão de inteiros (3ª versão)



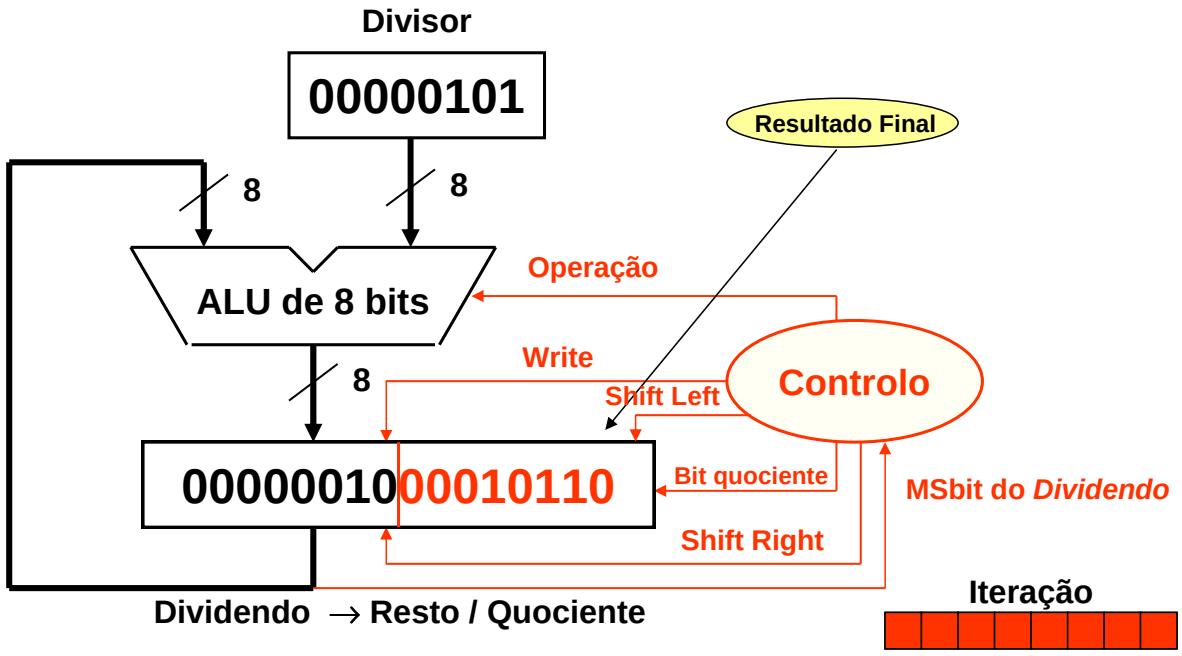
Arquitectura de um Divisor (versão final)



Na versão final do divisor, o registo ***Quociente*** desaparece, sendo substituído pela parte menos significativa do registo ***Dividendo/Resto***.

Arquitectura de um Divisor (versão final)

(exemplo c/ operandos de 8 bits: 01110000 ÷ 00000101=00010110, Resto=10)



Divisão de inteiros com sinal

A **divisão de inteiros com sinal** faz-se em **sinal e módulo**

Nas **divisões com sinal** aplicam-se as seguintes **regras**:

- Dividem-se dividendo por divisor em módulo
- O quociente terá sinal negativo se os sinais de dividendo e divisor forem diferentes
- O resto terá o mesmo sinal que o dividendo

Exemplos: **-7 / 3 = -2** c/ resto = **-1**

7 / -3 = -2 c/ resto = **1**

Dividendo = Divisor * Quociente + Resto

A Divisão de inteiros no MIPS

- No MIPS, a divisão é assegurada por uma arquitectura *hardware* semelhante à anteriormente descrita para a multiplicação (a unidade de controlo é que estabelece a diferença)
- Tal como acontecia na multiplicação, continua a existir a necessidade de um registo de 64 bits para armazenar o valor inicial do dividendo, e bem assim o resultado final na forma de um quociente e de um resto.
- Os mesmos registos, **HI** e **LO**, que tinham já sido apresentados para o caso da multiplicação, são igualmente utilizados para a divisão:
  - o registo **HI** armazena o **resto da divisão** inteira
  - o registo **LO** armazena o **quociente da divisão** inteira

A Divisão de inteiros no MIPS

Em *Assembly*, a divisão é efectuada pela instrução

```
div   $reg1, $reg2      # Divide (signed)
divu  $reg1, $reg2      # Divide unsigned
```

em que $reg1 é o dividendo e $reg2 o divisor. O **resultado** fica armazenado nos registos **HI (resto)** e **LO (quociente)**.

A **transferência** de informação entre os registos **HI e LO** e os restantes **registos de uso geral** faz-se através das instruções:

```
mfhi  $reg      # move from hi - Copia HI para $reg
mflo  $reg      # move from lo - Copia LO para $reg
mthi  $reg      # move to hi - Copia $reg para HI
mtlo  $reg      # move to lo - Copia $reg para LO
```