

Aula 12

Arquitectura de um divisor de números inteiros

Abordagem em três etapas

Exemplos de concretização

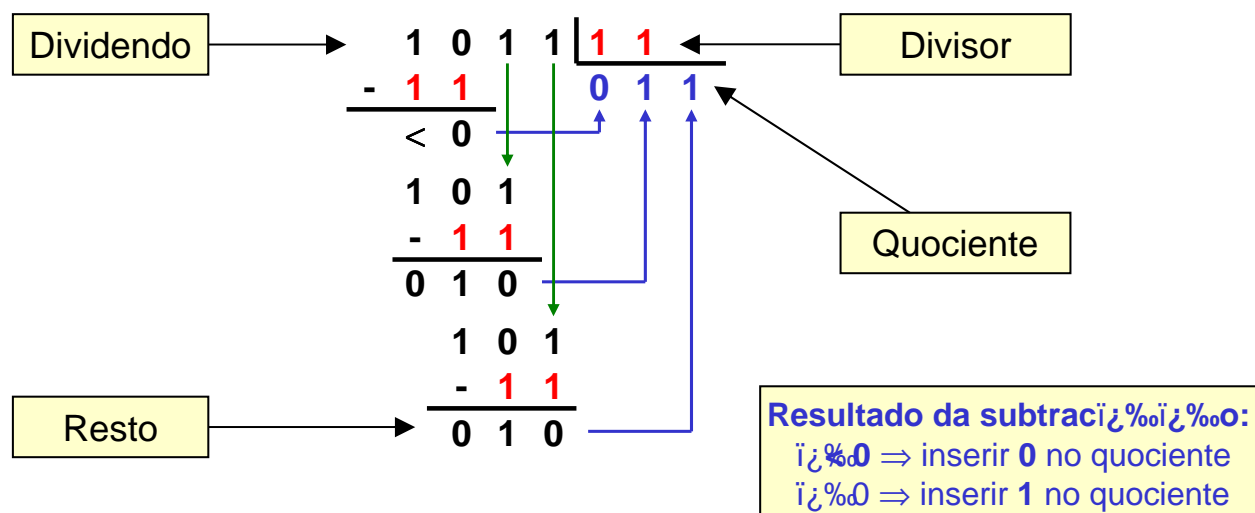
Divisão de inteiros com sinal

Divisão de inteiros no MIPS

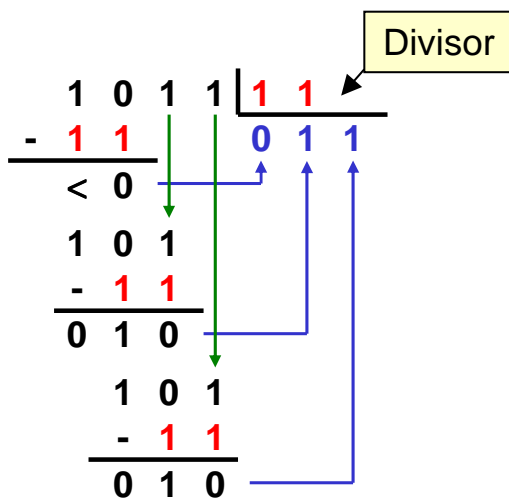
Bernardo Cunha, José Luís Azevedo, Arnaldo Oliveira e Silva

Divisão de inteiros em binário

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



Divis o de inteiros em bin rio



1. Come a-se por alinhar o Divisor   esquerda com o Dividendo

2. Subtrai-se o Divisor do Dividendo

  Se o resultado **positivo** (i.e. Dividendo \geq Divisor) acrescenta-se "1" no Quociente

  Se o resultado **negativo** (i.e. Dividendo $<$ Divisor) acrescenta-se "0" no Quociente e rep e-se o dividendo

3. Se o Divisor ainda n o est  alinhado   direita com o Dividendo, ent o desloca-se o Divisor 1 bit para a direita

4. Repete-se desde 2

  Como fazer o alinhamento do divisor   esquerda de forma m tica?

  Quantas itera es s o necess rias, no caso geral, para fazer a divis o?

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

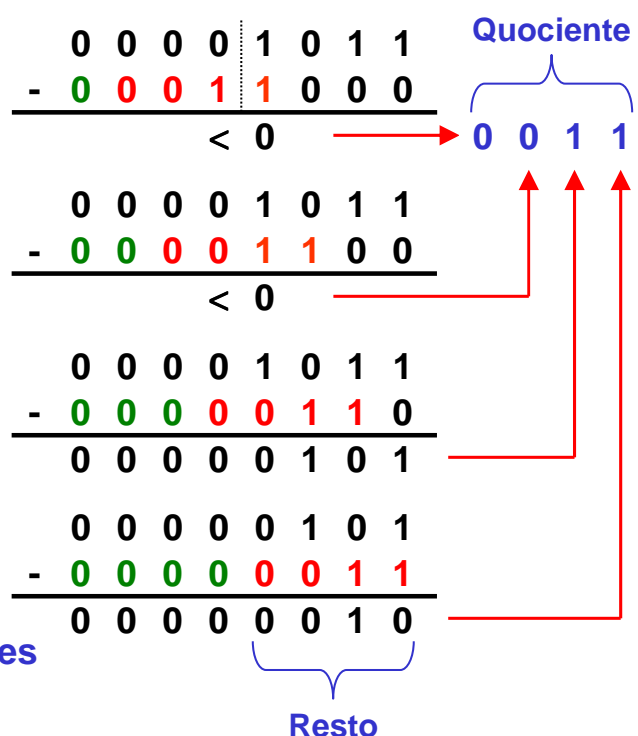
  Com operandos de 4 bits os registos para alojar o **dividendo** e o **divisor** t m 8 bits

  **Dividendo** alinhado **direita** (os 4 MSbits s o colocados a 0)

  **Divisor** alinhado **esquerda** (os 4 LSbits s o colocados a 0)

  Por cada nova itera o o **divisor**   **deslocado   direita 1 bit**

  **n mero total de itera es** igual ao **n mero de bits do dividendo original**



**Algoritmo para divisïo
de inteiros (1ï versïo)**



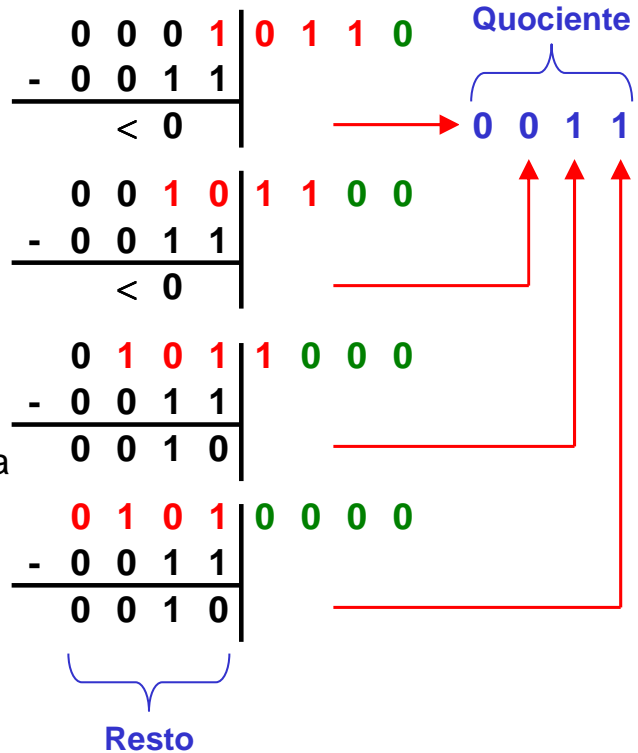
1 0 1 1 | 0 0 1 1

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-se assim a dimensão original (4 bits no exemplo)

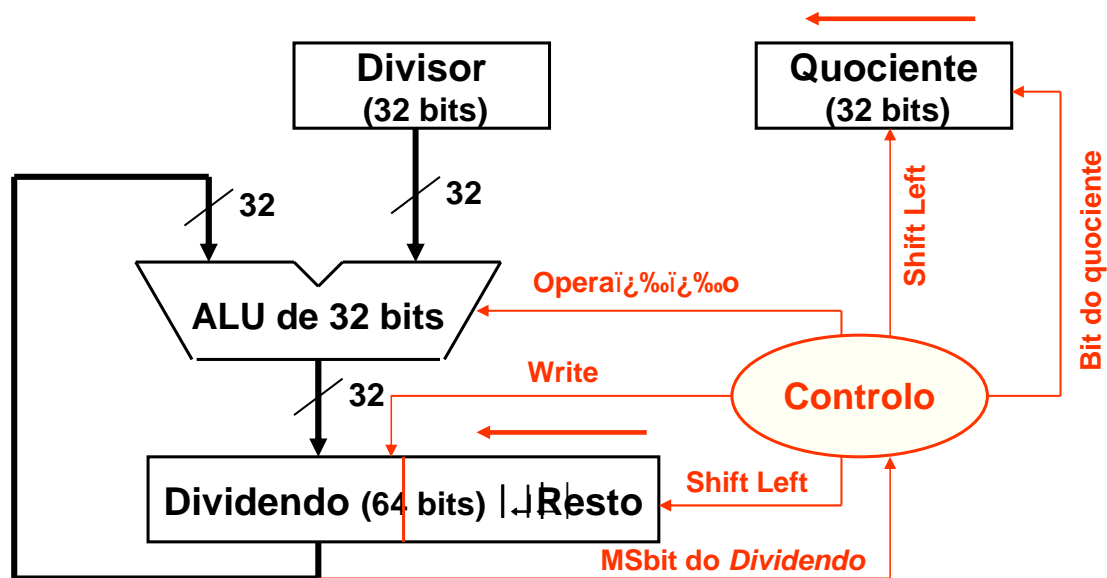
• A **subtracção** (entre dividendo e divisor) pode também ser feita apenas com **4 bits**, reduzindo-se para metade a dimensão da ALU.



Algoritmo para divisão de inteiros (2ª versão)



Arquitetura de um Divisor (2ª versão)



Nesta 2ª versão do divisor, o registro **Divisor** opera com 32 bits.

1 0 1 1 | 0 0 1 1

0 0 0 0 1 0 1 1 Dividendo

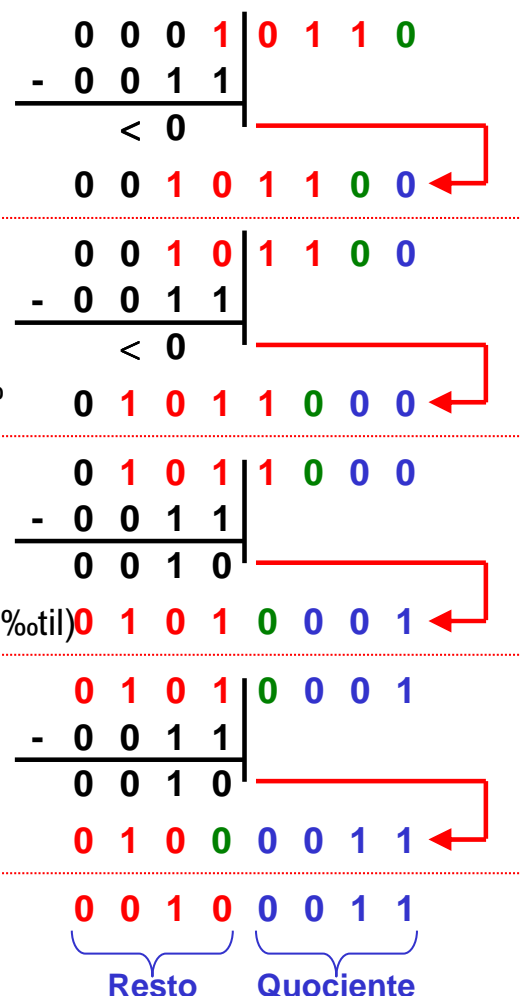
0 0 0 1 | 0 1 1 0 Dividendo após $\ll 1$
0 0 1 1 | Divisor

Pode verificar-se que o deslocamento à esquerda do conteúdo do **Dividendo**, é acompanhado por um deslocamento à esquerda do **Quociente**

Em cada deslocamento à esquerda do **Dividendo** é introduzido um zero (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

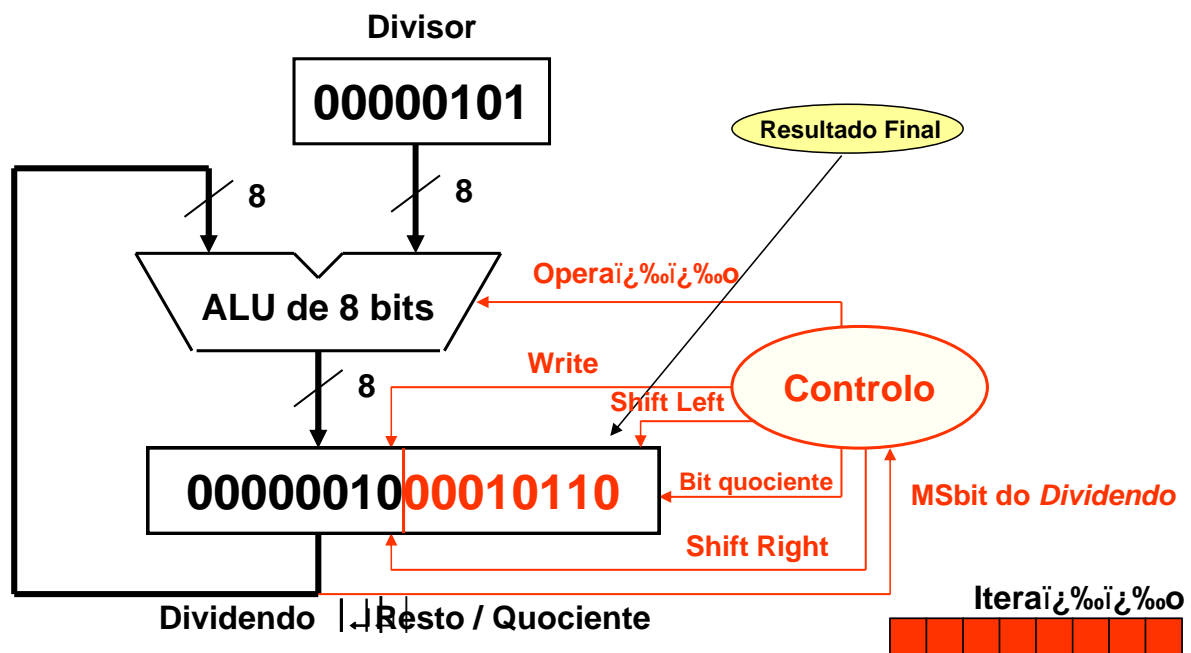


**Algoritmo para divisïo
de inteiros (3ï versïo)**



Arquitectura de um Divisor (versão final)

(exemplo c/ operandos de 8 bits: $01110000 \div 00000101 = 00010110$, 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 do 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

$$\text{Dividendo} = \text{Divisor} * \text{Quociente} + \text{Resto}$$

A Divis o de inteiros no MIPS

No MIPS, a divis o   assegurada por uma *architecture* semelhante   anteriormente descrita para a multiplic o (a unidade de controlo que estabelece a diferen a)

Tal como acontecia na multiplic o, ~~confirma 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 j  sido apresentados para o caso da multiplic o, s o igualmente utilizados para:

o registo **HI** armazena o **resto da divis o inteira**

o registo **LO** armazena o **quociente da divis o inteira**

A Divis o de inteiros no MIPS

Em *Assembly*, a divis o   efectuada pela instrui o

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

em que \$reg1   o dividendo e \$reg2 o divisor. **Resultado** fica armazenado nos registos **HI (resto)** e **LO (quociente)**.

A **transfer ncia** de informa o entre os registos **HI** e **LO** e os restantes **registos de uso geral** faz-se atrav s das instrui es:

```
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
```