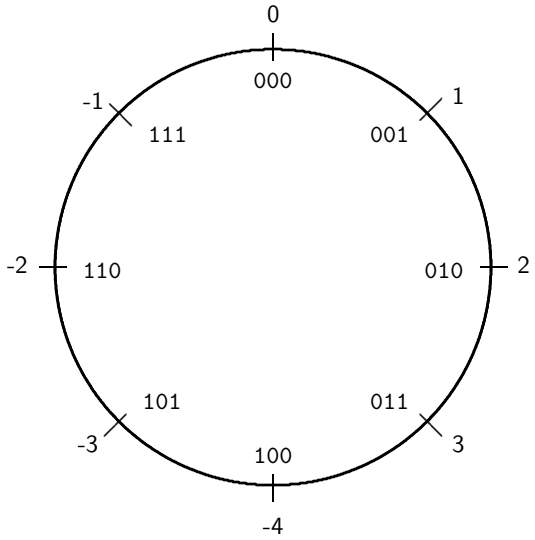
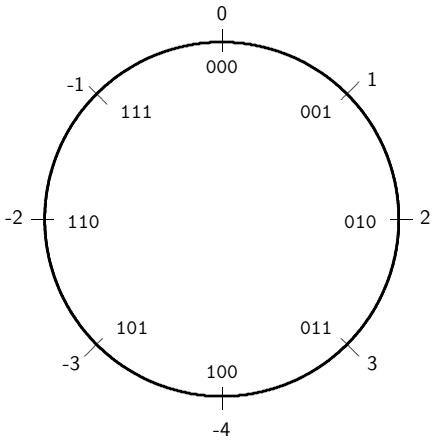
Representação em Ponto Fixo – Complemento de 2



Representação em Ponto Fixo (cont)

Representação em complemento de dois
círculo da representação + overflow

$$
\begin{aligned}
x + \overline{x} &= -1 \\
x + \overline{x} + 1 &= 0 \\
\overline{x} + 1 &= -x
\end{aligned}
$$



Representação em Ponto Fixo (cont.)

$$
\begin{aligned}
(x_{31} \cdot -2^{31}) &+ [\, x_{30} \cdot 2^{30} + \ldots + x_0 \cdot 2^0 \,] \\
(0 \cdot -2^{31}) &+ [\ldots] \quad \rightarrow 0001 \\
(1 \cdot -2^{31}) &+ [\ldots] \quad \rightarrow 1111
\end{aligned}
$$

$$
\begin{aligned}
(0 \cdot -2^2) + 1 \cdot 2^1 + 1 \cdot 2^0 &= (x_2 \cdot -2^2) + x_1 \cdot 2^1 + x_0 \cdot 2^0 \\
0 + 2 + 1 = 3 &= (0 \cdot -2^2) + 1 \cdot 2^1 + 0 \cdot 2^0
\end{aligned}
$$

$$
\begin{aligned}
(1 \cdot -2^2) + 1 \cdot 2^1 + 1 \cdot 2^0 &= (x_2 \cdot -2^2) + x_1 \cdot 2^1 + x_0 \cdot 2^0 \\
-4 + 2 + 1 = -1 &= (x_2 \cdot -2^2) + x_1 \cdot 2^1 + x_0 \cdot 2^0
\end{aligned}
$$

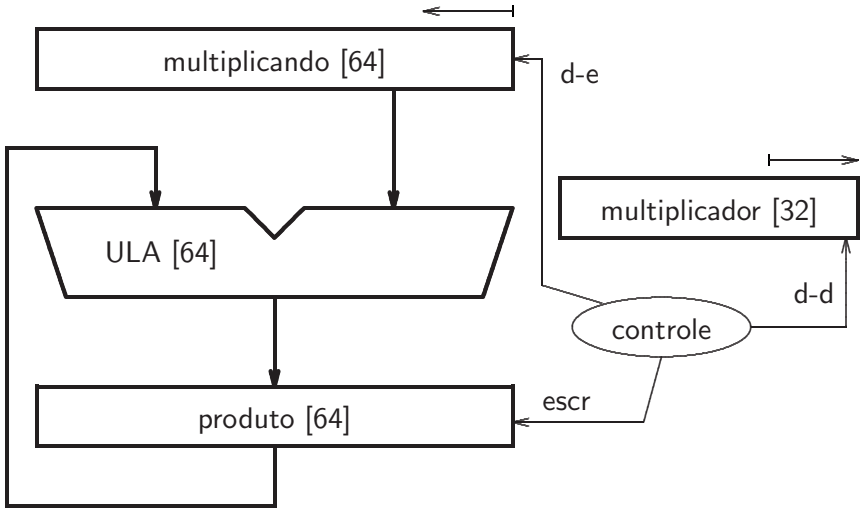
Multiplicação

Efetuar $\mathbf{1000}_{10} \times \mathbf{1001}_{10}$

conclusões?

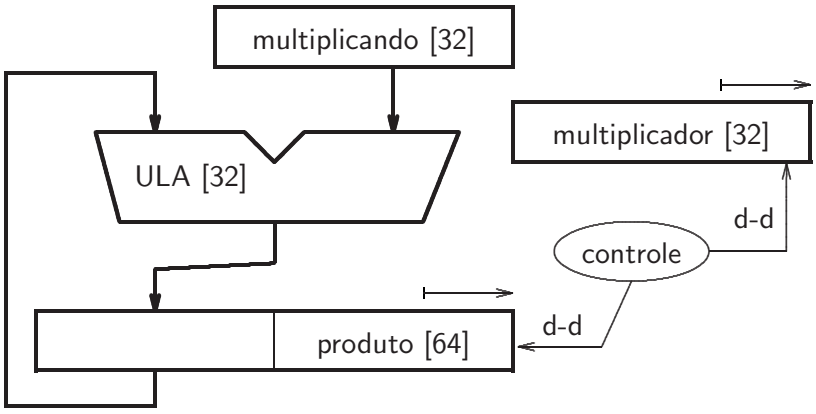
```
for (i=0; i < 32 ; i++) {
   if (multiplicador & 01 == 1)
      produto += multiplicando ;
   multiplicando << 1 ;
   multiplicador >> 1 ;
}
```


Circuito multiplicador V1



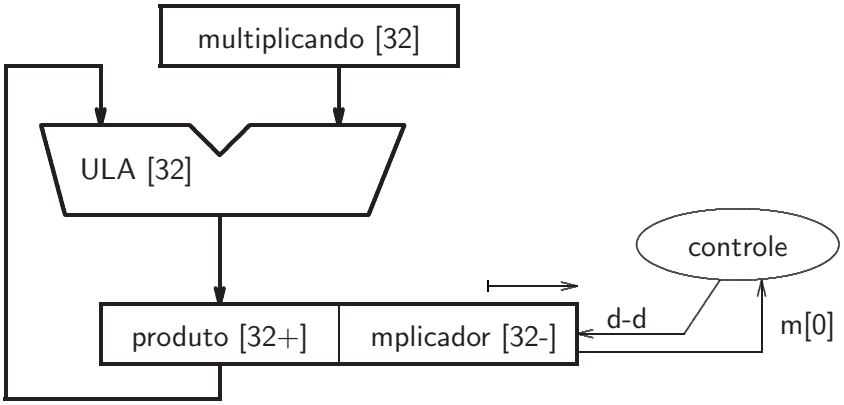
Circuito multiplicador V2

Metade do multiplicando é sempre zero,
e bits menSign do produto não mudam após soma
→ deslocar produto para a direita!



Circuito multiplicador V3

Problema: iniciam 32 bits menos significativos do produto em zero; multiplicador desloca-se para a direita...



Circuito multiplicador V3

Sinais: negativo se sinais diferentes;
converte negativos para positivos e lembra sinais

Otimização:

- pular dois zeros de cada vez
- desenrolar loop
- Algoritmo de Booth – leitura e lista!
- ver questão sobre multiplicador de 2005-2 (primeira prova)

Divisão

Efetuar $\mathbf{100\ 1010_{10}/1000_{10}}$

conclusões?

Divisão

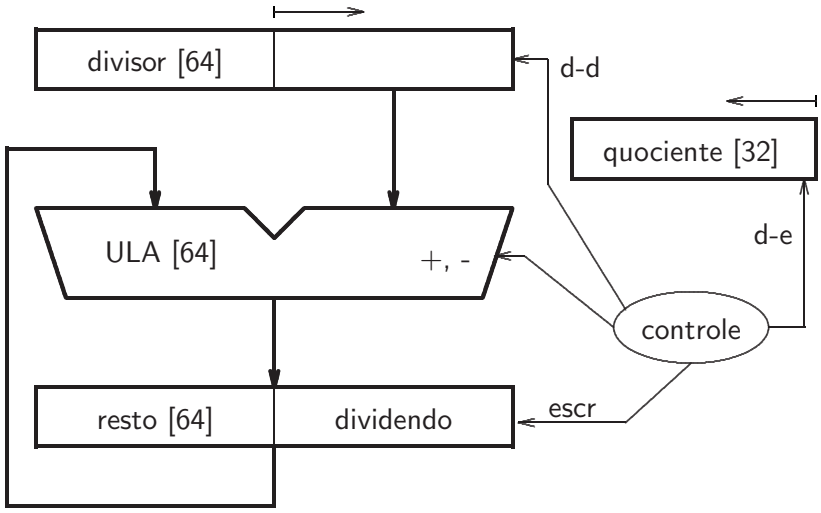
$$\text{dividendo} = \text{quociente} * \text{divisor} + \text{resto}$$

```
resto = dividendo ;
for (i=0; i < 33 ; i++) {
   resto -= divisor;
   if (resto >= 0) {
      quociente = (quociente << 1) + 1 ;
   } else {
      resto += divisor ;
      quociente = (quociente << 1) + 0 ;
   }
   divisor >> 1 ;
}
```


Circuito divisor V1

```
/* INIC: divisor na metade esquerda de divisor  */
/* e dividendo na metade direita de resto  */
resto = dividendo ;
for (i=0; i < 33 ; i++) {
   resto -= divisor;
   if (resto >= 0) {
      quociente = (quociente << 1) + 1 ;
   } else {
      resto += divisor ;
      quociente = (quociente << 1) + 0 ;
   }
   divisor >> 1 ;
}
```


Circuito divisor V1

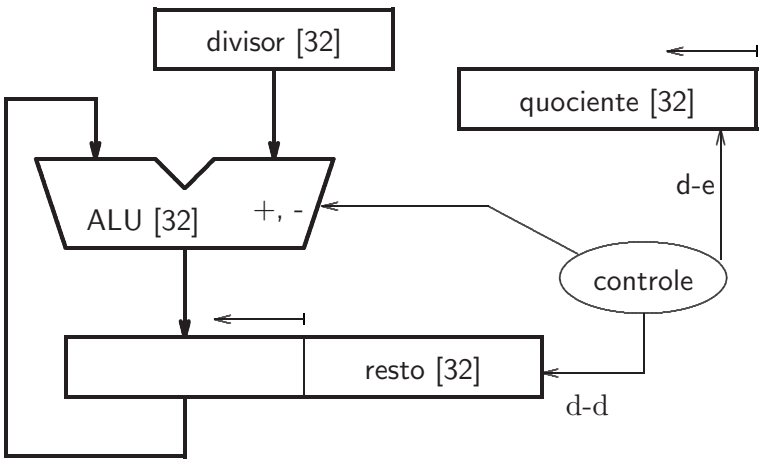


Circuito divisor V2

Problema: metade do divisor é inútil;
`resto` pode ser deslocado para esquerda: 32 iterações

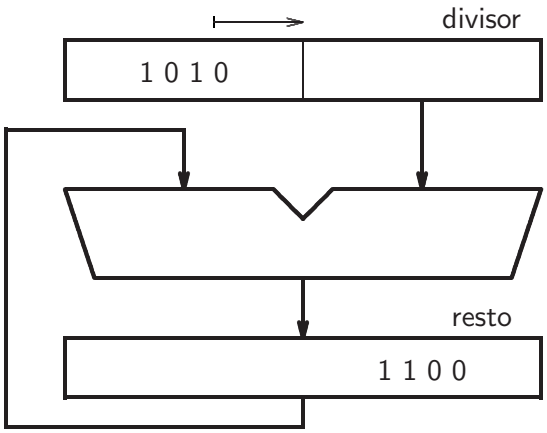
```
/* INIC: dividendo na direita de resto  */
resto = dividendo ;
for (i=0; i < 32 ; i++) {
   resto = resto << 1 ;
   resto -= divisor;
   if (resto >= 0) {
      quociente = (quociente << 1) + 1 ;
   } else {
      resto += divisor ;
      quociente = (quociente << 1) + 0 ;
   }
}
```


Circuito divisor V2



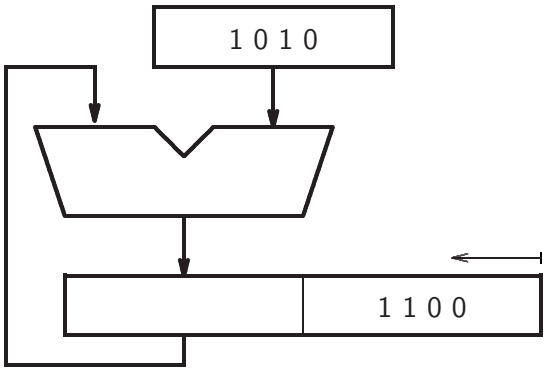
Divisor V1 *vs* V2

Diferença no número de passos dos algoritmos:
V1: primeira subtração é SEMPRE zero!



Divisor V1 *vs* V2

V2: se deslocar resto ANTES de subtrair,
economiza primeiro passo do algoritmo da versão V1

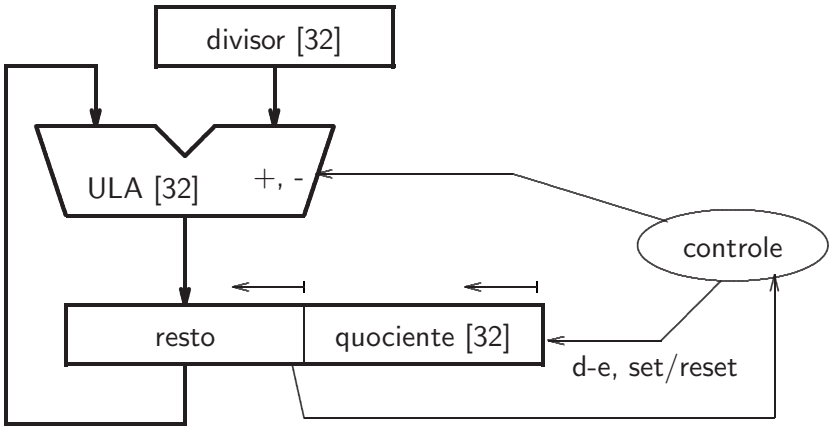


Circuito divisor V3

Inserir bits do quociente em `dividendo/resto`
soma somente 32 bits mais significativos do `dividendo/resto`

```
resto = dividendo ;
for (i=0; i < 32 ; i++) {
   resto = resto << 1 ;
   resto -= divisor;
   if (resto >= 0) {
      (resto = resto << 1) + 1 ;
   } else {
      resto += divisor ;
      (resto = resto << 1) + 0 ;
   }
}
[(resto = resto >> 1)63:32] ;
```


Circuito divisor V3



Divisão: etc

Sinais na divisão:
resto tem sinal do dividendo
quociente negativo se sinais diferentes

Implementação:
mesmo circuito multiplica e divide!

Optimizações?