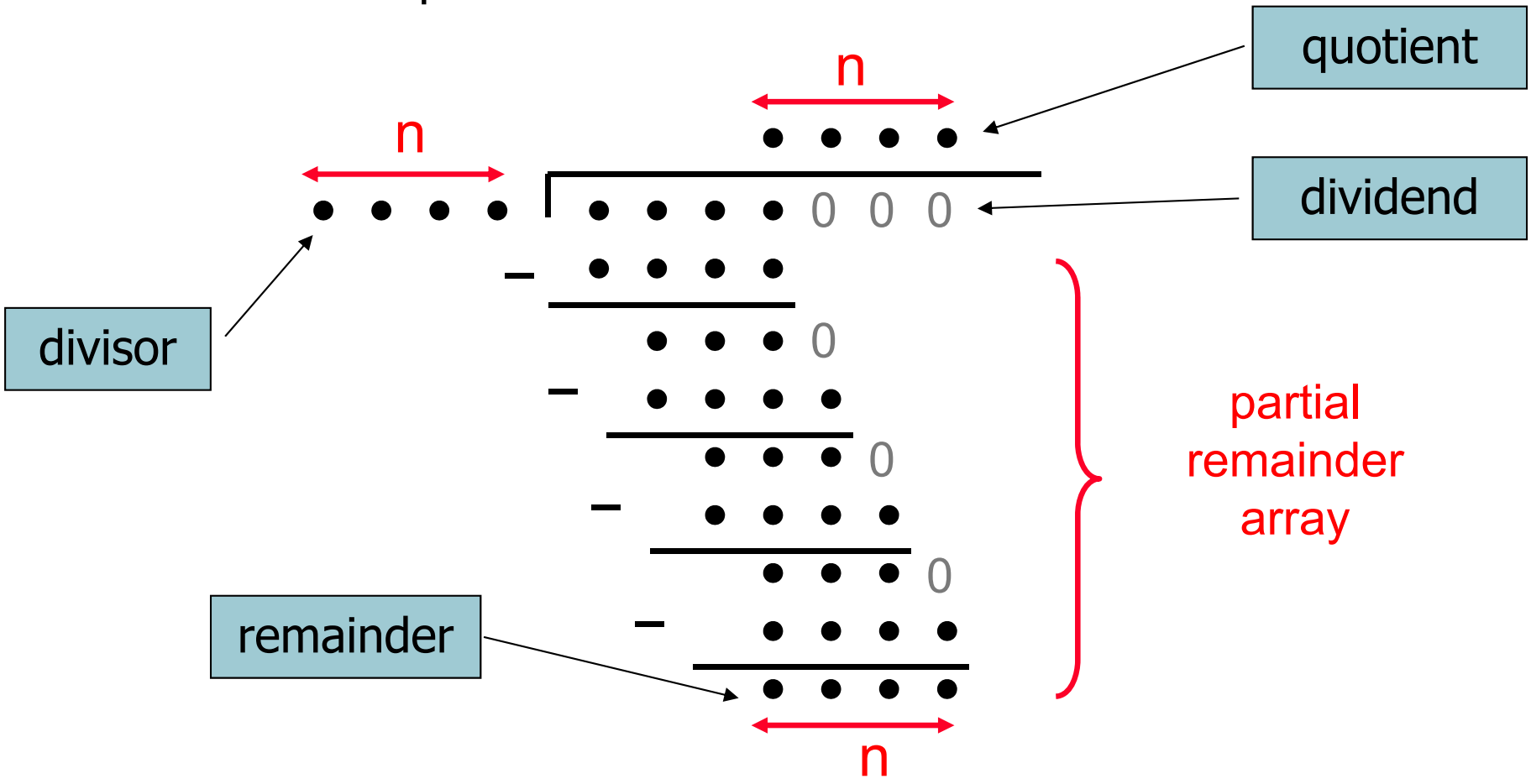
Lecture 8
Division

School of Computer Science and Engineering
Soongsil University

3. Arithmetic for Computers

3.4 Division

- Division is just a bunch of quotient digit guesses and right shifts and subtracts
  - dividend = quotient x divisor + remainder



Pencil and Paper Algorithm

- $1001010_{ten} \div 1000_{ten}$

```
                        1 0 0 1 ten      (quotient)
(Divisor) 1 0 0 0 ten | 1 0 0 1 0 1 0 ten  (dividend)
                    - 1 0 0 0
                      ---------
                      0 0 0 1
                        0 0 1 0
                          0 1 0 1
                            1 0 1 0
                          - 1 0 0 0
                          ---------
                            0 0 1 0 ten  (remainder)
```

A Division Algorithm and Hardware

- **First Version - Hardware**

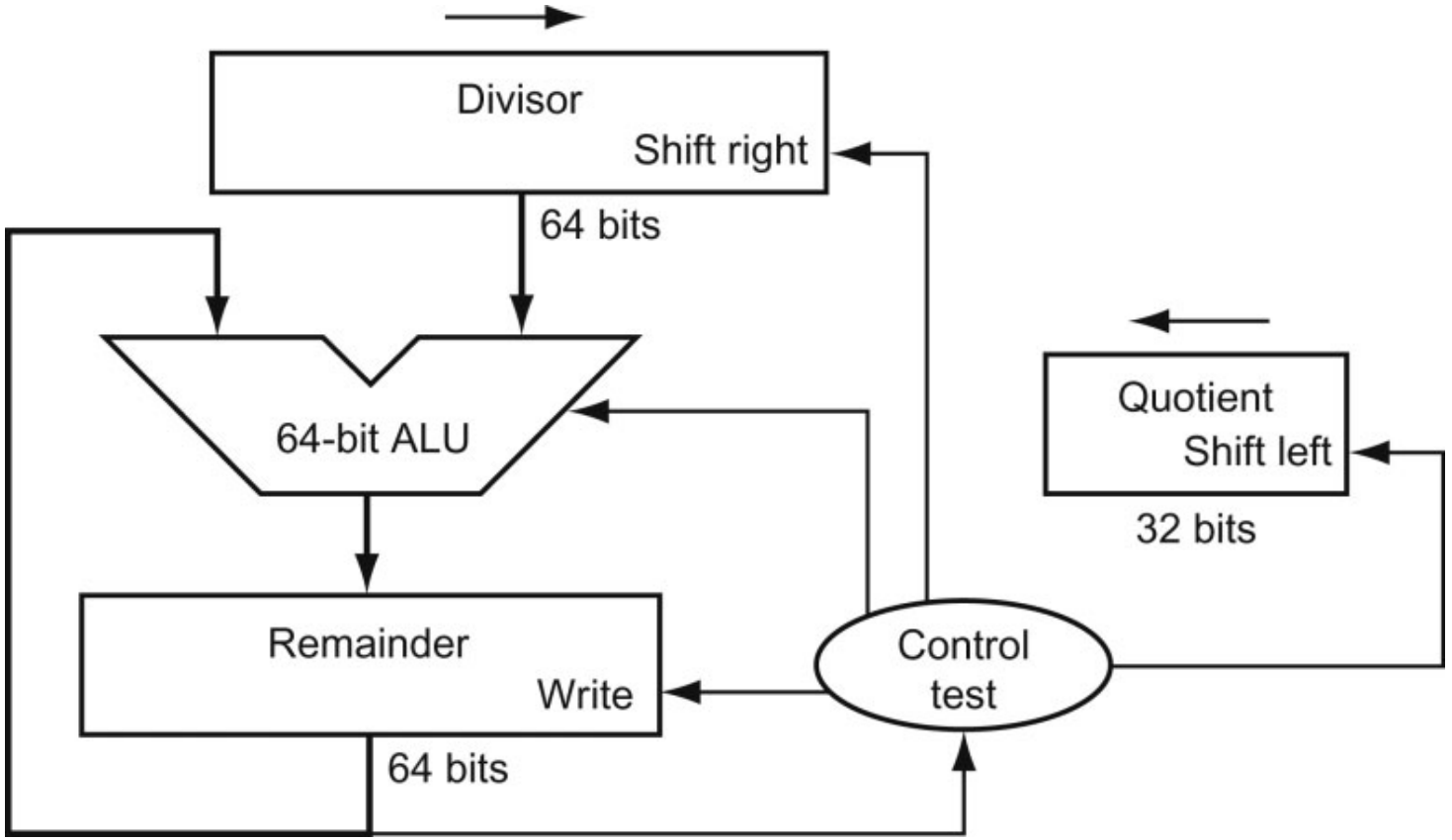


Figure 3.8

First Version - Algorithm

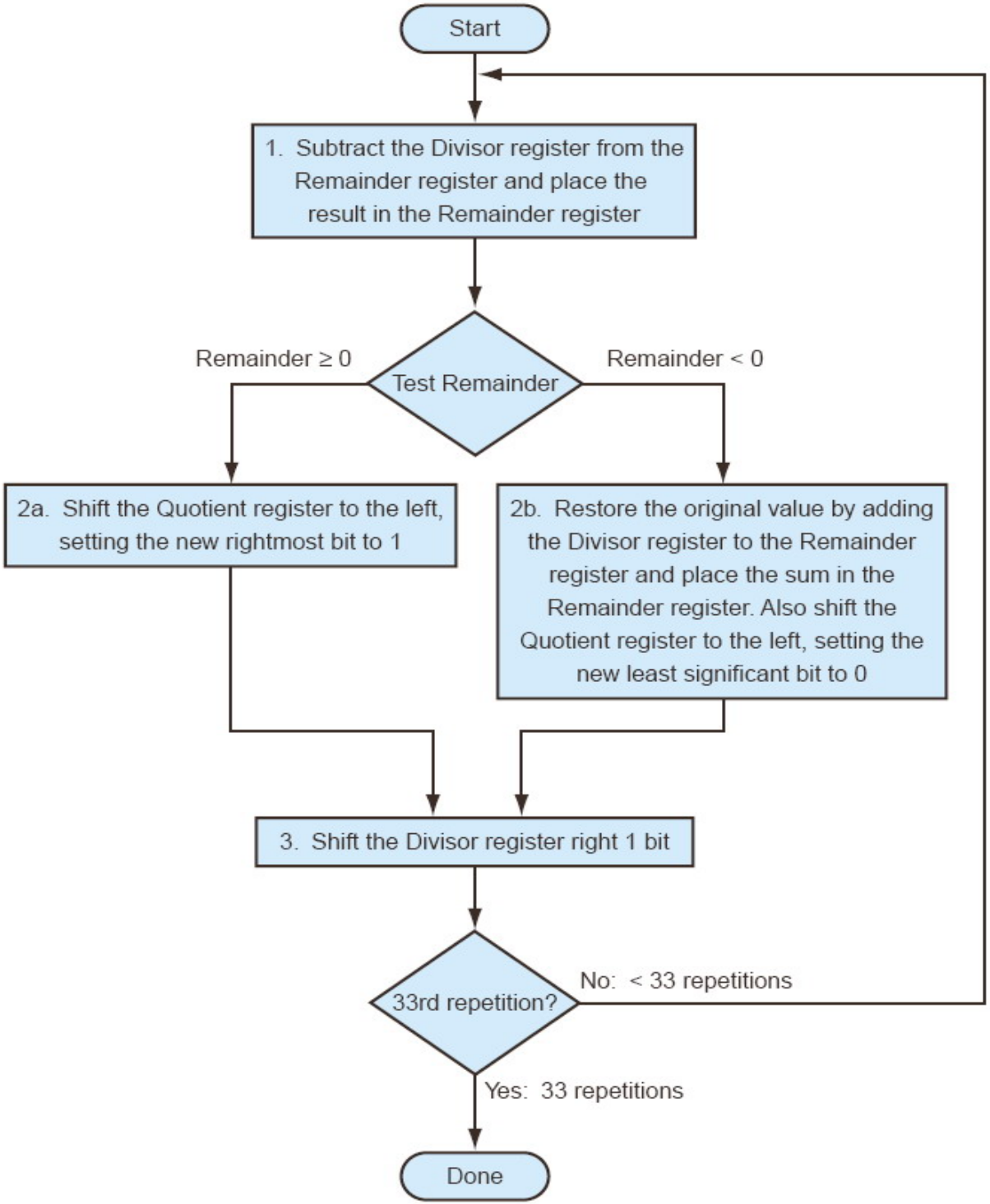


Figure 3.9

Example: Divide $0000\ 0111_{two}$ by 0010_{two}.

Iteration	Step	Quotient	Divisor	Remainder
0	Initial values	0000	0010 0000	0000 0111
1	1: Rem = Rem - Div			1110 0111
	2b: Rem<0 ⇒ +Div, sll Q, Q_0=0	0000		0000 0111
	3: Shift Div right		0001 0000	
2	1: Rem = Rem - Div			1111 0111
	2b: Rem<0 ⇒ +Div, sll Q, Q_0=0	0000		0000 0111
	3: Shift Div right		0000 1000	
3	1: Rem = Rem - Div			1111 1111
	2b: Rem<0 ⇒ +Div, sll Q, Q_0=0	0000		0000 0111
	3: Shift Div right		0000 0100	
4	1: Rem = Rem - Div			0000 0011
	2a: Rem≥0 ⇒ sll Q, Q_0=1	0001		
	3: Shift Div right		0000 0010	
5	1: Rem = Rem - Div			0000 0001
	2a: Rem≥0 ⇒ sll Q, Q_0=1	0011		
	3: Shift Div right		0000 0001	0000 0001

Figure 3.10

Second Version - Hardware

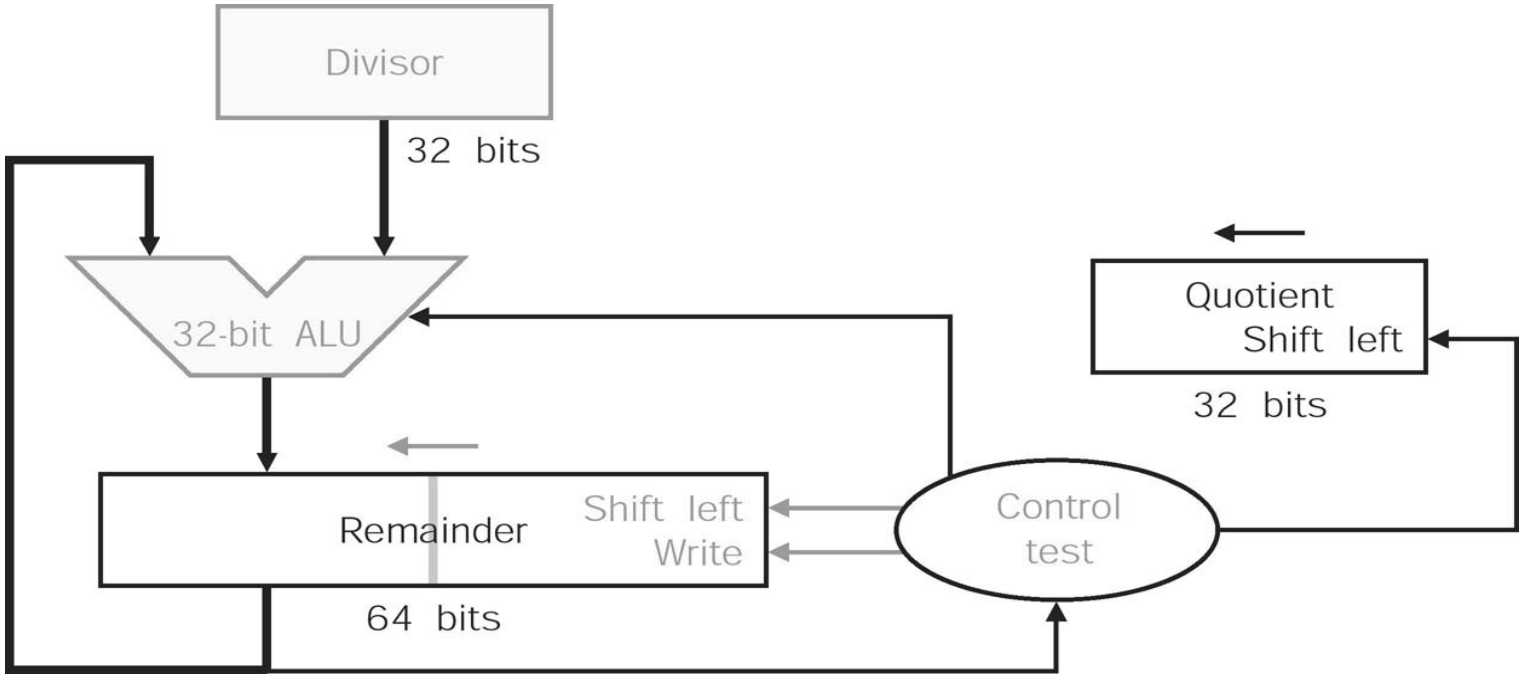


Figure 4.39 in 2ed

Improved Version - Hardware

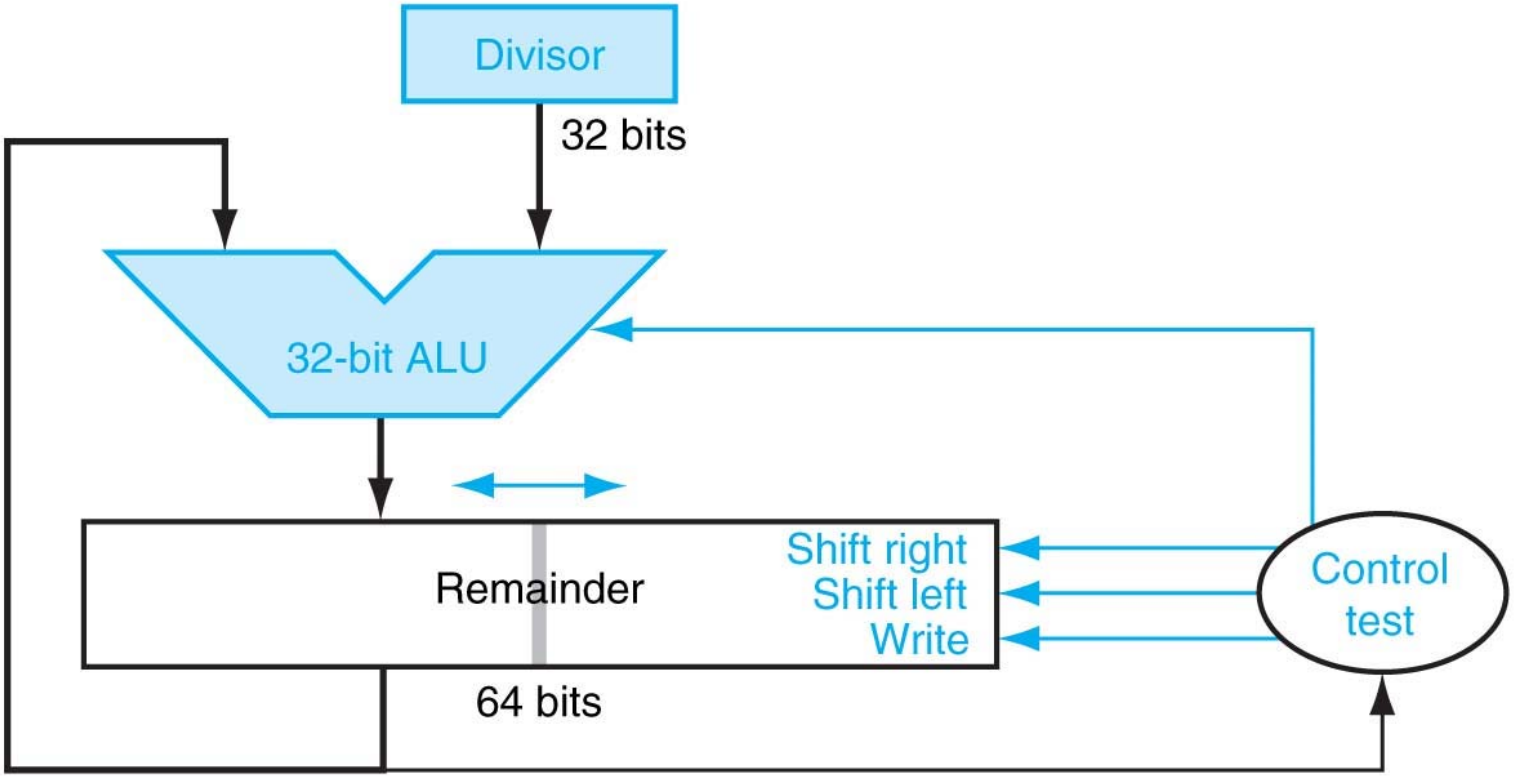


Figure 3.11

Improved Version - Algorithm

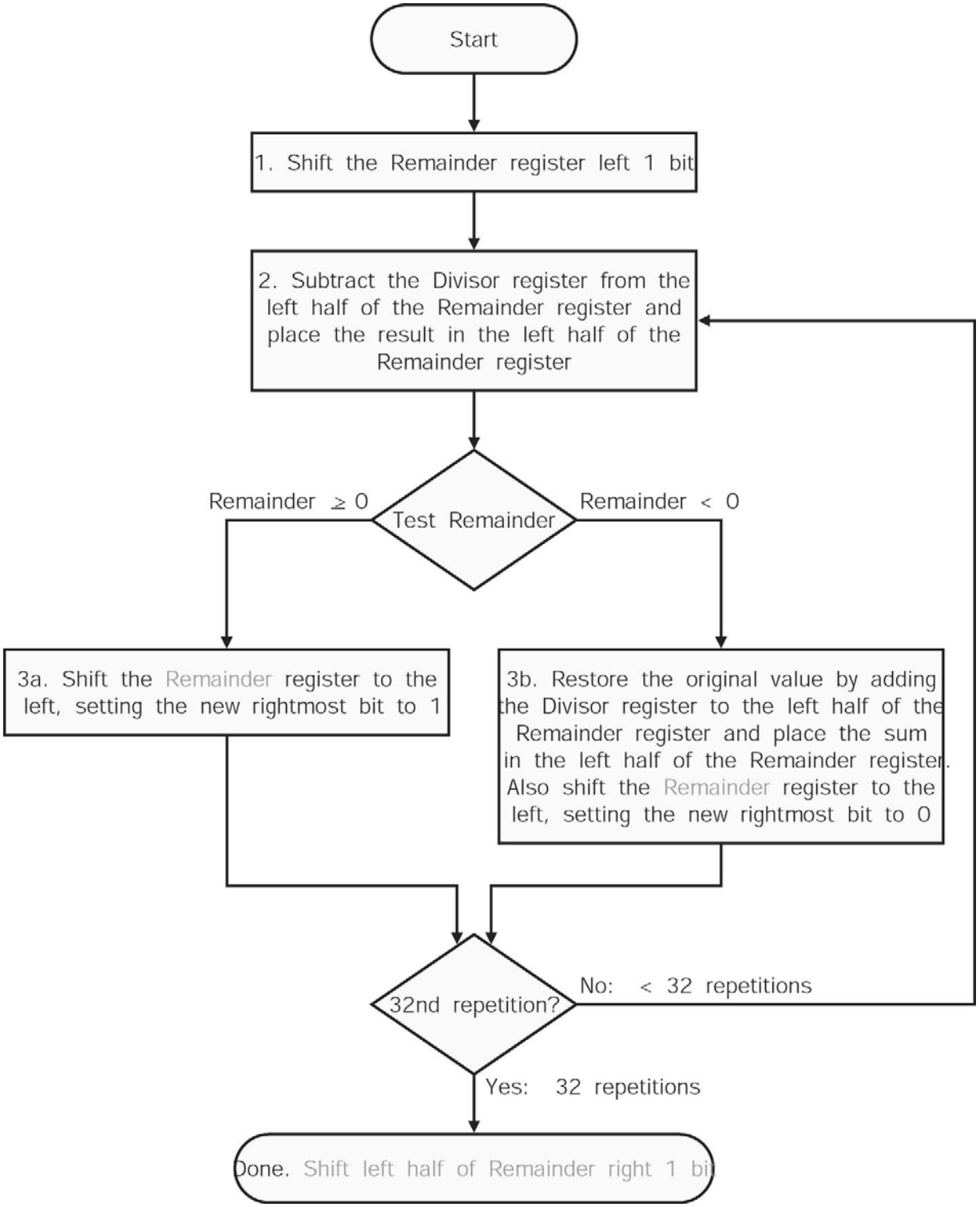


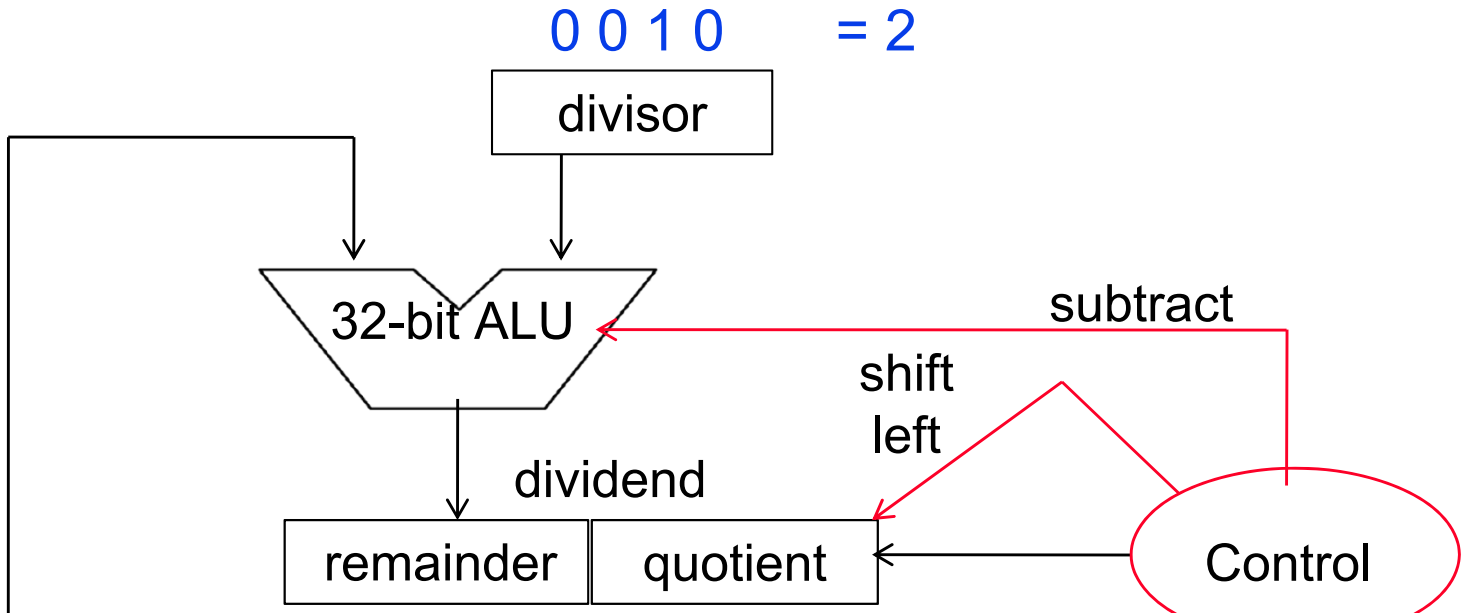
Figure 4.40 in 2ed.

Example 1: Improved Version

- **Divide $0000\ 0111_{two}$ by 0010_{two}.**

Iteration	Step	Divisor	Remainder
0	Initial values	0010	0000 0111
	Shift Rem left		0000 1110
1	2: Rem = Rem - Div		1110 1110
	3b: Rem<0 ⇒ +Div, sll R, R_0=0		0001 1100
2	2: Rem = Rem - Div		1111 1100
	3b: Rem<0 ⇒ +Div, sll R, R_0=0		0011 1000
3	2: Rem = Rem - Div		0001 1000
	3a: Rem≥0 ⇒ sll R, R_0=1		0011 0001
4	2: Rem = Rem - Div		0001 0001
	3a: Rem≥0 ⇒ sll R, R_0=1		0010 0011
	Shift left half of Rem right 1		0001 0011

Example 2: Improved Version



	0 0 0 0	0 1 1 0	= 6
	0 0 0 0 ←	1 1 0 0	
sub	1 1 1 0	1 1 0 0	rem < 0, so Q bit = 0
	0 0 0 0	1 1 0 0	restore remainder
	0 0 0 1 ←	1 0 0 0	
sub	1 1 1 1	1 0 0 0	rem < 0, so Q bit = 0
	0 0 0 1	1 0 0 0	restore remainder
	0 0 1 1 ←	0 0 0 0	
sub	0 0 0 1	0 0 0 1	rem ≥ 0, so Q bit = 1
	0 0 1 0 ←	0 0 1 0	
sub	0 0 0 0	0 0 1 1	rem ≥ 0, so Q bit = 1
			= 3 with 0 remainder

Signed Division

- **Sign of the quotient**
  - Minus when the signs of the divisor and dividend disagree
- **Sign of the remainder**
  - Same as the sign of the dividend

Divide in MIPS

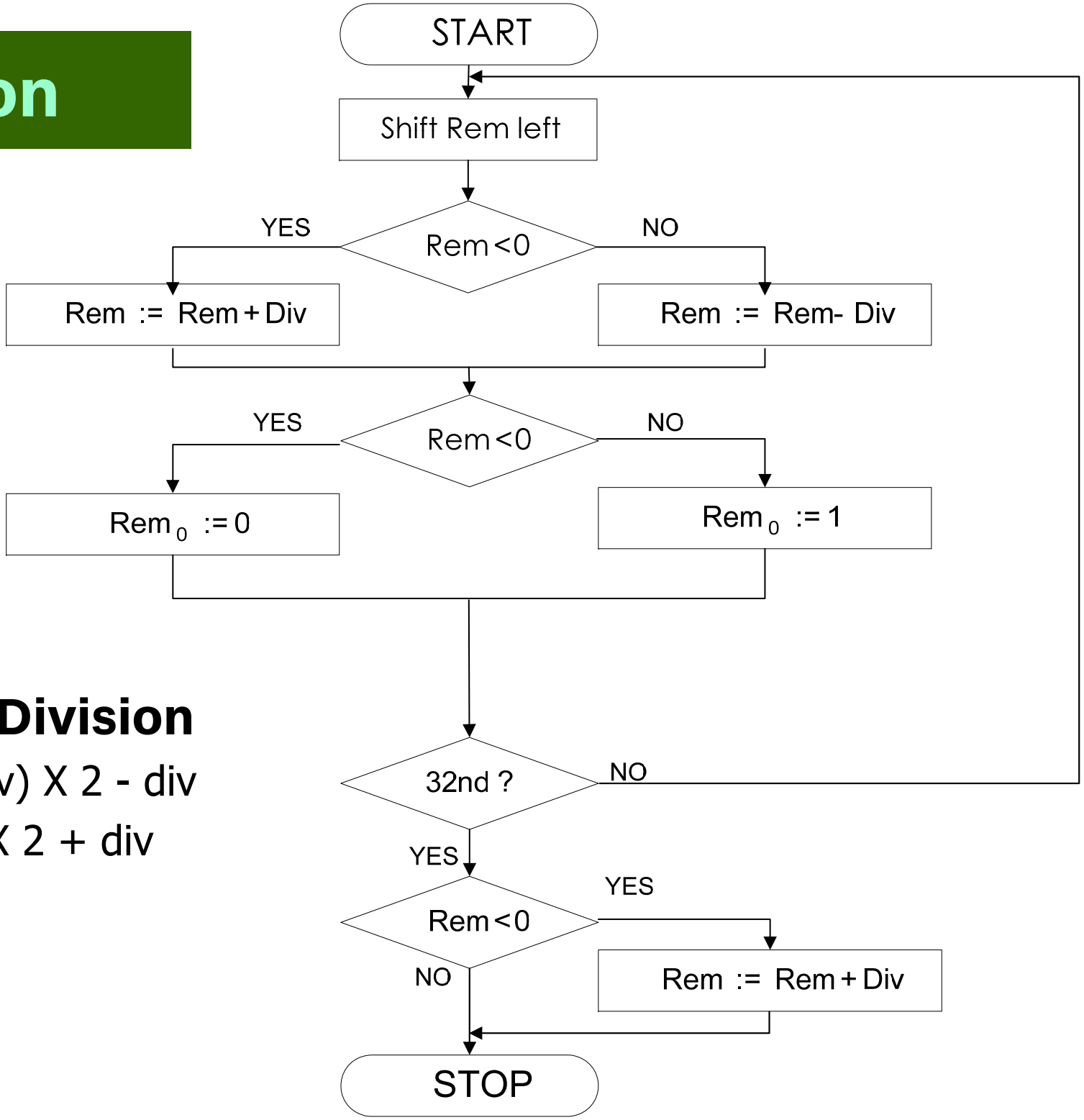
- Divide (**`div`** and **`divu`**) generates the reminder in **`hi`** and the quotient in **`lo`**

```
div    $s0, $s1      # lo = $s0 / $s1
                     # hi = $s0 mod $s1
```

0	16	17	0	0	0x1A

  - Instructions **`mfhi rd`** and **`mflo rd`** are provided to move the quotient and reminder to (user accessible) registers in the register file
- As with multiply, divide ignores overflow so software must determine if the quotient is too large. Software must also check the divisor to avoid division by 0.

Elaboration



- **Nonrestoring Division**
  - (rem - div + div) X 2 - div
    = (rem - div) X 2 + div

Example: Nonrestoring Division

- **00001 10001 (=49) ÷ 00101 (=5)**

Iteration	Step	Divisor	Remainder
0	Initial values	00101	00001 10001
1	Shift Rem left		00011 00010
	Rem≥0 ⇒ Rem = Rem - Div		11110 00010
	Rem<0 ⇒ R_0=0		11110 00010
2	Shift Rem left		11100 00100
	Rem<0 ⇒ Rem = Rem + Div		00001 00100
	Rem≥0 ⇒ R_0=1		00001 00101
3	Shift Rem left		00010 01010
	Rem≥0 ⇒ Rem = Rem - Div		11101 01010
	Rem<0 ⇒ R_0=0		11101 01010
4	Shift Rem left		11010 10100
	Rem<0 ⇒ Rem = Rem + Div		11111 10100
	Rem<0 ⇒ R_0=0		11111 10100
5	Shift Rem left		11111 01000
	Rem<0 ⇒ Rem = Rem + Div		00100 01000
	Rem≥0 ⇒ R_0=1		00100 01001

MIPS Arithmetic Instructions

Category	Instruction	Example		Meaning	Comments
Arithmetic	add	add	$s1,$s2,$s3	$s1 = $s2 + $s3	Three operands; overflow detected
	subtract	sub	$s1,$s2,$s3	$s1 = $s2 – $s3	Three operands; overflow detected
	add immediate	addi	$s1,$s2,100	$s1 = $s2 + 100	+ constant; overflow detected
	add unsigned	addu	$s1,$s2,$s3	$s1 = $s2 + $s3	Three operands; overflow undetected
	subtract unsigned	subu	$s1,$s2,$s3	$s1 = $s2 – $s3	Three operands; overflow undetected
	add immediate unsigned	addiu	$s1,$s2,100	$s1 = $s2 + 100	+ constant; overflow undetected
	move from coprocessor register	mfc0	$s1,$epc	$s1 = $epc	Copy Exception PC + special regs
	multiply	mult	$s2,$s3	Hi, Lo = $s2 × $s3	64-bit signed product in Hi, Lo
	multiply unsigned	multu	$s2,$s3	Hi, Lo = $s2 × $s3	64-bit unsigned product in Hi, Lo
	divide	div	$s2,$s3	Lo = $s2 / $s3, Hi = $s2 mod $s3	Lo = quotient, Hi = remainder
	divide unsigned	divu	$s2,$s3	Lo = $s2 / $s3, Hi = $s2 mod $s3	Unsigned quotient and remainder
	move from Hi	mfhi	$s1	$s1 = Hi	Used to get copy of Hi
	move from Lo	mflo	$s1	$s1 = Lo	Used to get copy of Lo

Figure 3.12