CS470
Chap 4: Computer Arithmetic
Multiply/Divide

Text: Patterson Hennessey

Yashwant K. Malaiya

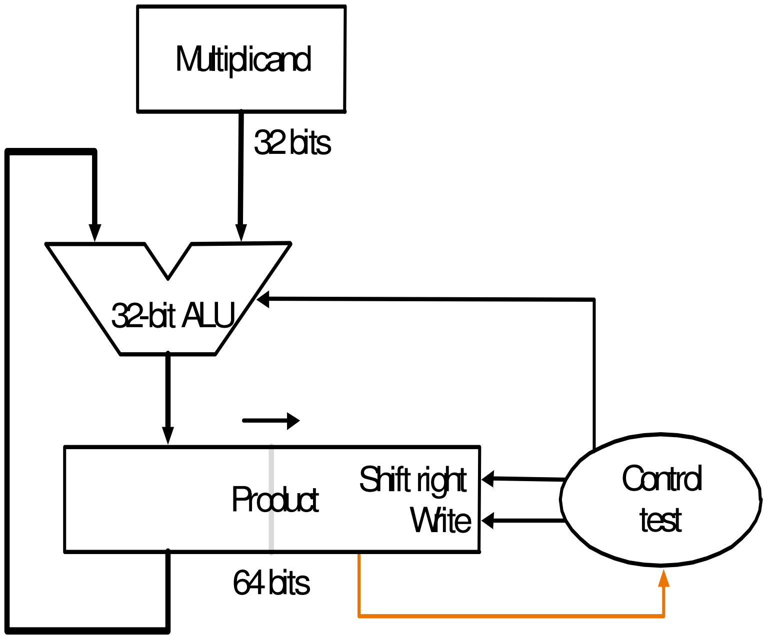
Multiplication

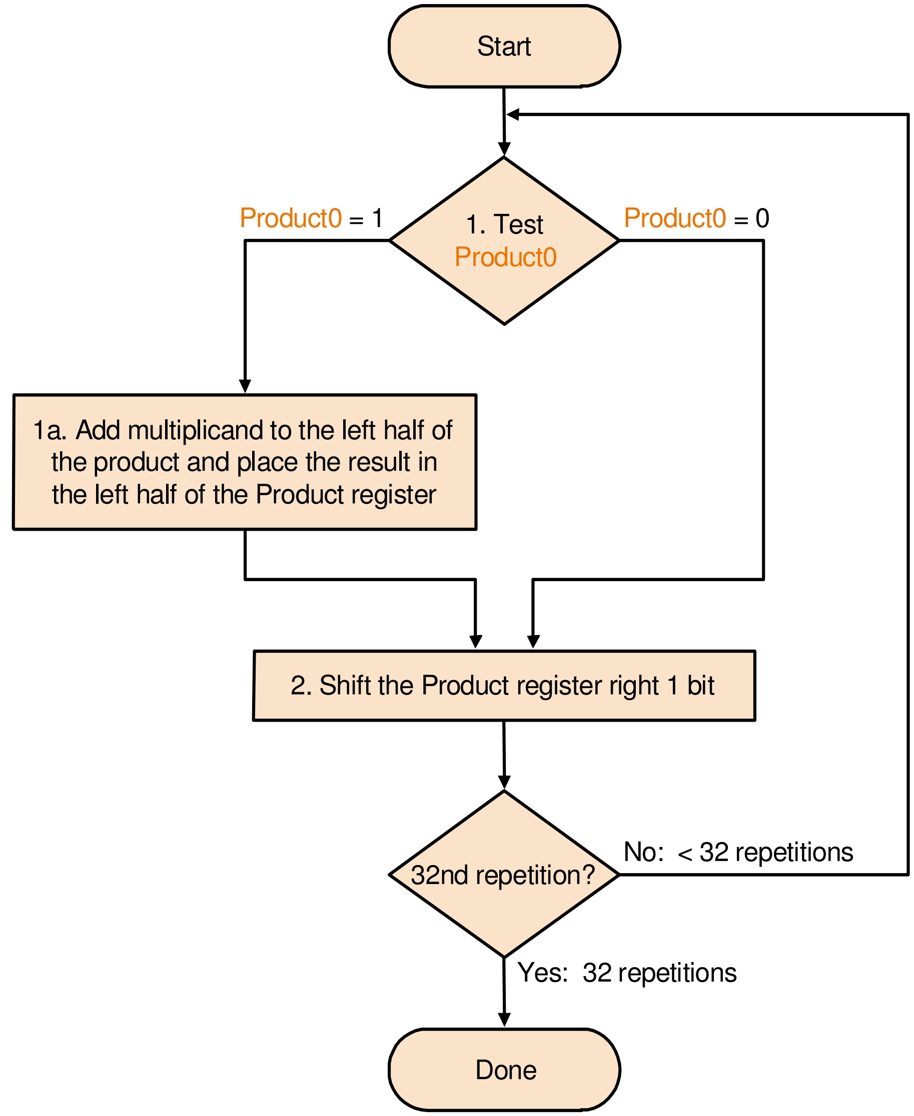
- **More complicated than addition**
  - accomplished via shifting and addition
- **More time and more area**
- **Let's look at 3 versions based on gradeschool algorithm**

```
      0010   (multiplicand)
   x  1011   (multiplier)
```

- **Negative numbers: convert and multiply**
  - there are better techniques, we won't look at them

Multiplication





Multiplication

			0010 Mcand
iteration	**Step**	**Multiplicand**	**Product**
0	Initial values	0010	0000 0011 multiplier
1	1=>Prod=Prod+Mcand		0010 0011
	Shift right product		0001 0001
2	1=>Prod=Prod+Mcand		0011 0001
	Shift right product		0001 1000
3	0=>no op		0001 1000
	Shift right product		0000 1100
4	0=>no op		0000 1100
	Shift right product		0000 0110

Product

Note multiplier bits are colored differently

Multiplication: comments

- **Improvement: an extra bit on the left to hold a carry, incoming bit 0**
- **Special designs for efficient signed multiplication: Booth's multiplier**

Integer division

- **Pencil and paper binary division (decimal numbers)**

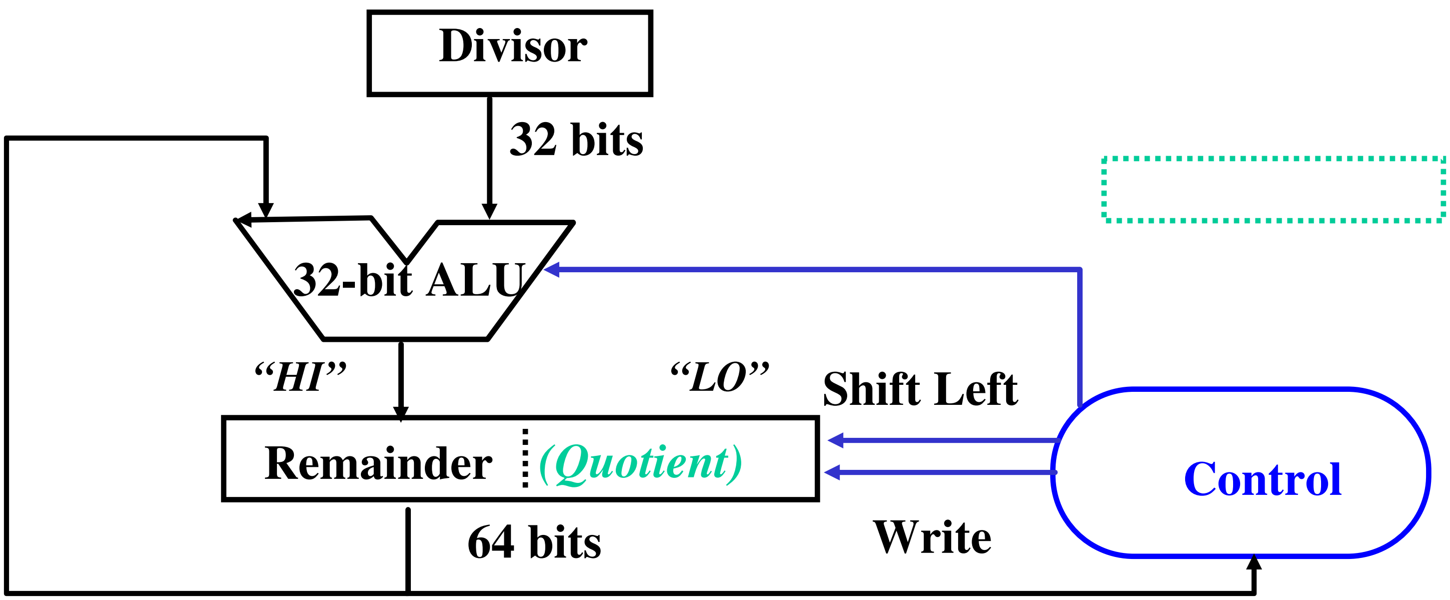
```
                        1001      (quotient)
(divisor)  1000 |    1001000      (dividend)
                    - 1000
                   --------
                     0001000
                    - 0001000
                   ----------
                        0000      (remainder)
```

- **Steps in hardware**
  - Shift the dividend left one position
  - Subtract the divisor from the left half of the dividend
  - If result positive, shift left a 1 into the quotient
  - Else, shift left a 0 into the quotient, and repeat from the beginning
  - Once the result is positive, repeat the process for the partial remainder
  - Do n iterations where n is the size of the divisor
- **Specific implementation:** Place partial remainder & Quotient in same reg.

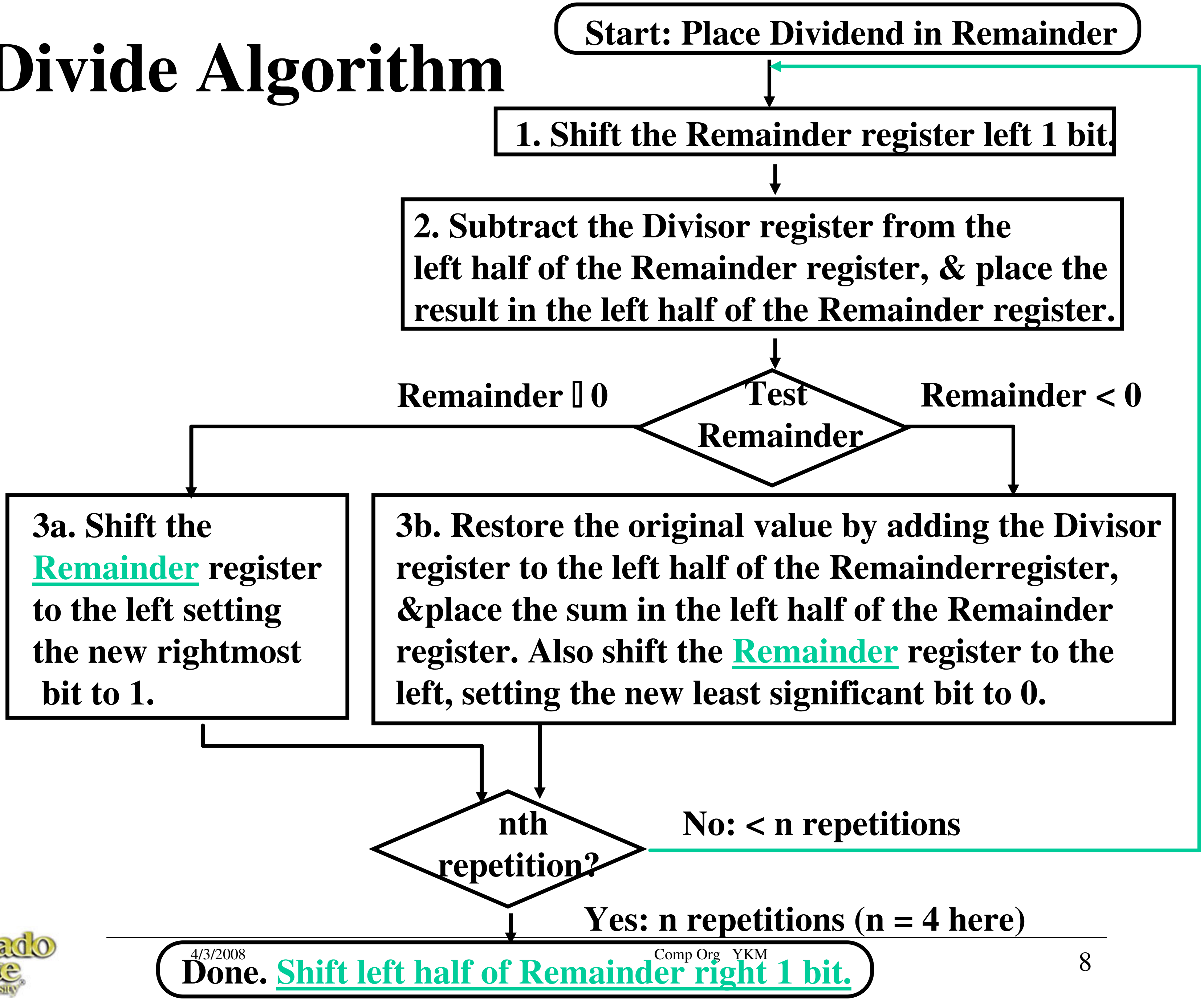
Integer division

- **Hardware implementation**

- **32-bit Divisor reg, 32 -bit ALU, 64-bit Remainder reg, (<u>No separate</u> Quotient reg)**



Divide Algorithm



Division: divide 0000 0111 by 0010

	"restoring division"		**0010 Divisor** **1110 Negated**
iteration	**Step**	**Divisor**	**Remainder**
0	Initial values	0010	0000 1111
	Shift Rem left 1		0000 1110
1	Rem = Rem -Div		1110 1110
	Rem<0 => + Div, sll R, R0 = 0		0001 1100
2	Rem = Rem -Div		1111 1100
	Rem<0 => + Div, sll R, R0 = 0		0011 1000
3	Rem = Rem -Div		0001 1000
	Rem≥ 0 => sll R, R0 = 1		0011 0001
4	Rem = Rem -Div		0001 0001
	Rem≥ 0 => sll R, R0 = 1		0010 0011
	Shift left half of Rem right 1		0001 0011

Note quotient bits are colored differently

Rem Quo

Division

- Other improved implementations are possible.
- Non-restoring division is more efficient.

Multiply and Divide in MIPS

- **32 bit Hi and 32-bit Lo registers**
- **Mflo, mfhi: move from lo, hi**
- **Multiplication: result in Hi-Lo**
  - Mult rs, rt, multu rs, rt
  - Mul d, s1, s2   3-register psuedoinstruction
- **Division: at end- Hi: remainder, Lo: quotient**
  - Div rs, rt; divu rs, rt
  - Div rd, rs1, rs2   psuedoinstruction: 3 registers