Computer Arithmetic
Division

Chapter 3.5
EEC170 FQ 2005

Division Terms

- Consider a long division example:

```
                0005671 ten   <- Quotient
Divisor  1234 ten | 7006789 ten   <- Dividend
                  7006
                 -6170
                  8367   <- Partial Remainder
                 -7404
                  9638
                 -8638
                  10009
                 -9872
                   137 ten   <- Remainder
```


Human Division

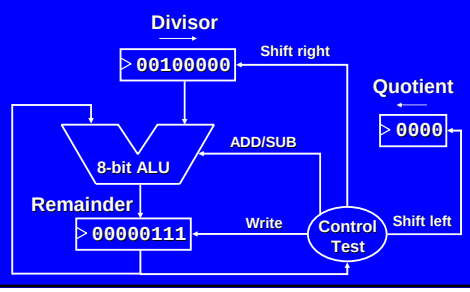
1. Bring down next digit to Partial Remainder
2. Compare Divisor and Partial Remainder
   - IF Partial Remainder < Divisor: put 0 digit in Quotient, goto 1.
3. Estimate Quotient digit
4. Multiply Divisor by estimated digit
5. Subtract from Partial Remainder
   - IF result is negative, erase work, decrement estimated digit, goto 4
   - IF result > Divisor, erase work, increment estimated digit, goto 4
   - ELSE put estimated digit in Quotient, goto 1

Binary Division

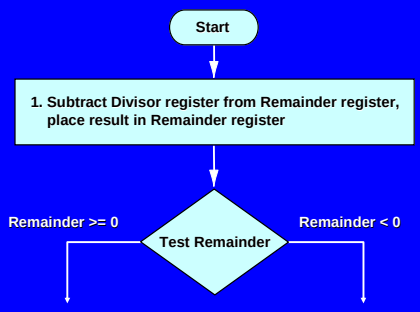
1. Bring down next digit to Partial Remainder
2. Subtract Divisor from Partial Remainder
   # "Estimate" Quotient bit will be 1
   - IF result is negative, erase work, put 0 in Quotient, goto 1
   - ELSE put 1 in Quotient, goto 1

Simple Hardware Divider

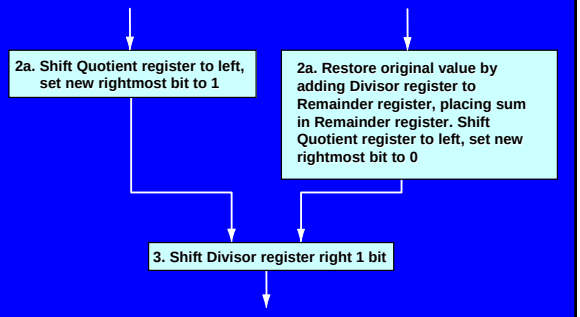
- Use ALU to subtract/add shifted Divisor from Remainder
  - how to initialize?
  - Conditionally add to 'erase' work after subtract



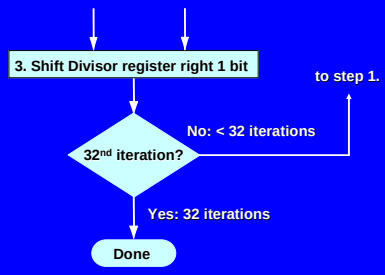
Simple Division Algorithm
Step 1



Simple Division Algorithm
Steps 2 & 3

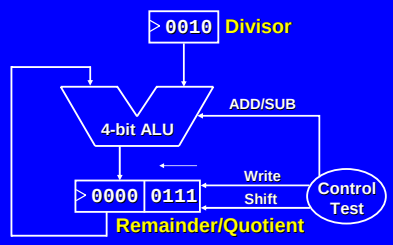


Simple Division Algorithm
Loop

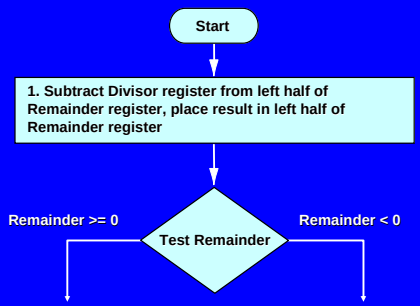


Improved Hardware Divider

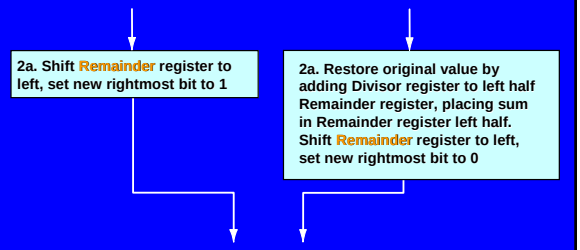
- Divisor, ALU are reduced in half
- Remainder shifts left, Divisor fixed
- Remainder/Quotient share register



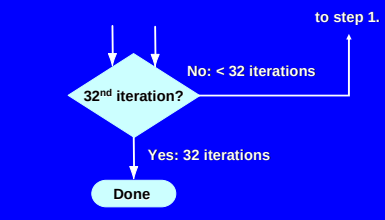
Improved Division Algorithm
Step 1



Improved Division Algorithm
Steps 2 & 3



Improved Division Algorithm
Loop



Signed Division

- Sign of Quotient is XOR of signs of Dividend and divisor
- Sign of Remainder matches sign of dividend

122 / 3 = 40 remainder 2

-122 / -3 = 40 remainder –2

-122 / 3 = -40 remainder –2

122 / -3 = -40 remainder 2

‘Fast’ Division

- So far all ALU operations can be done in $\log_2 N$ time units
  - Carry Look-Ahead Adder: 1 clock cycle
  - Wallace Tree Multiplier: 2 clock cycles
  - Barrel Shifter: 1 clock cycle
- Best division algorithms take N time units (slow)
  - Resolve 4 bits per iteration rather than 1, and iterations are complex
    - Intel floating point bug
  - 20 to 40 cycles for MIPS R10000