① 二進位除法運算

```
          10000   ← Quotient
      ┌──────────
1000  │ 10000100  ← Dividend
      │ 1000
      ──────
        100       ← Remainder
↑
Divisor
```

Dividend = divisor × quotient + remainder

模擬除法行為：

a. divisor 右移

b. 判斷能否能 dividend 相減

if 可 then quotient 的 bit 為 1

不可 = 0

c. quotient向左移

② 傳統除法器

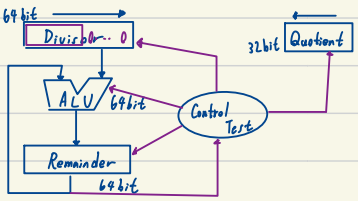
1. 確認需要元件：

(1). Quotient, divisor, remainder 暫存器

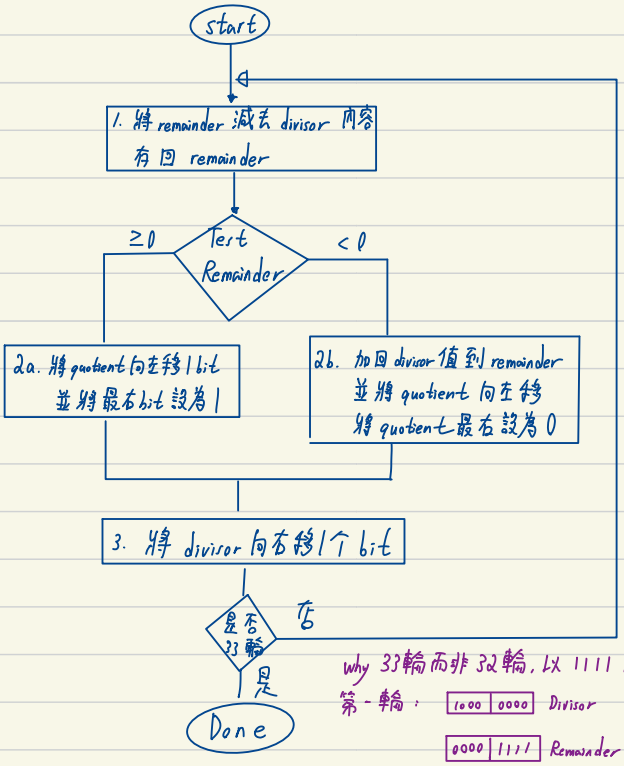
(2). ALU 作減, 加法

(3). Control test

2. 硬件



3. Control Test flow chart



why 33輪而非 32輪，以 1111 除 1000 為例

第一輪：

第一輪白做的

4. 計算題

例 $7_{(10)}$ 除以 $2_{(10)}$，用傳統除法器模擬

$7_{(10)} = 0111_{(2)}$ $2_{(10)} = 0010_{(2)}$ ⇒ 4個 bit：5輪

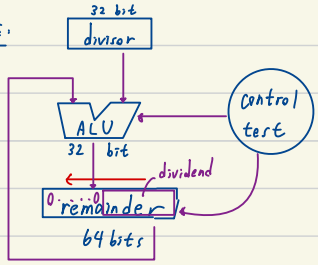
Iteration	Step	Quotient	Divisor	Remainder
0	initialize	0000	0010 0000	0000 0111
1	remainder ← remainder − divisor	0000	0010 0000	1110 0111
	remainder < 0 remainder + divisor quotient 左移 最右 bit 為 0	0000	0010 0000	0000 0111
	divisor 向右移	0000	0001 0000	0000 0111
⋮	⋮	⋮	⋮	⋮

③ Hardware-friendly 除法器

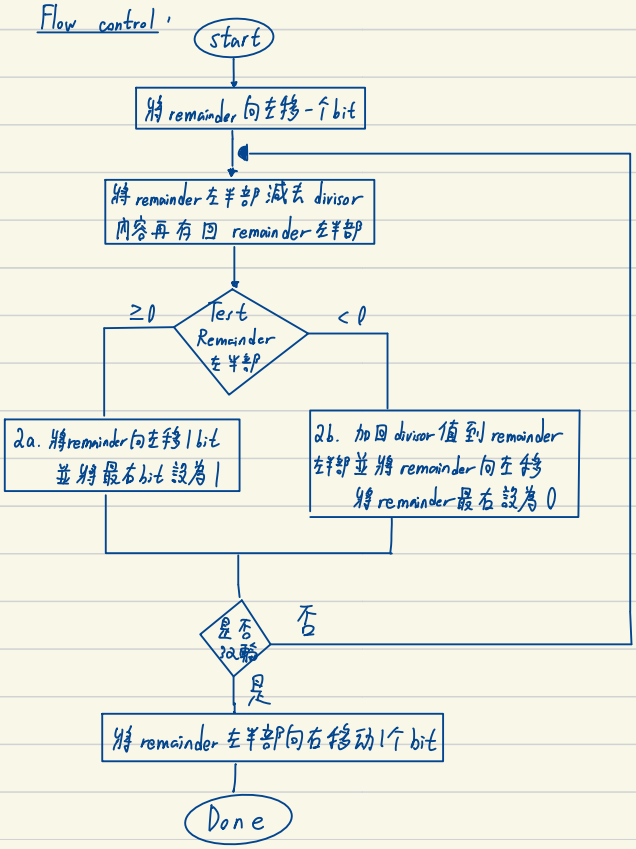
不移動 divisor ⇒ 移動 dividend (向左移)

∵ Quotient 和 Dividend 都向左移 ⇒ 一起存放在 Remainder Register

硬件：



Flow control：



☆ 為了不做到33輪，一開始 init 時就將 remainder 向左移

做完時，∵ remainder 會多向左移1位元，需向右移回

e.g. 以 $0111_{(2)}$ 除以 0010 為例

1. 確認的需暫存器：remainder (8 bit)，初始內容
   divisor (4 bit)

2. 的需輪數：4

iteration	step	divisor	remainder
0	init	0010	0000 0111
	remainder ←	0010	0000 1110
1	L(r) ← L(r)-d	0010	1110 1110
	<0, restore	0010	0000 1110
	remainder ← remainder 最右為0	0010	0001 1100
2	L(r) ← L(r)-d	0010	1111 1100
	<0, restore	0010	0001 1100
	remainder ← remainder 最右為0	0010	0011 1000
3	L(r) ← L(r)-d	0010	0001 1000
	>0, remainder ← remainder 最右為1	0010	0011 0001
4	L(r) ← L(r)-d	0010	0001 0001
	>0, remainder ← remainder 最右為1	0010	0010 0011
	L(r) →	0010	0001 0011

1 3 ∴ $7 = 2 \times 3 + 1$ check!

③. 除法器比較 & 流程圖

substract → check → action → shift

÷	Traditional		Hardware friendly	
prepare			$\overleftarrow{R}$	
(1). subtract	$R \leftarrow R - D$		$L(R) \leftarrow L(R) - D$	
(2) check	$R \geq 0$?			
(3). Action	Yes	No	Yes	No
	$\overleftarrow{Q} \leftarrow 1$	Restore $\overleftarrow{Q} \leftarrow 0$	$\overleftarrow{R} \leftarrow 1$	Restore $\overleftarrow{R} \leftarrow 0$
(4). Shift	$\vec{D}$			
			$\overrightarrow{L(R)}$	
32 bit 輪數	eg. 32 bit 33 輪		32 輪	

④. MIPS 中乘除法器硬体是一致的

∵ Hardware-friendly 的需暫存器數目及大小是相同的

但 Product or Remainder register 要可以左、右移

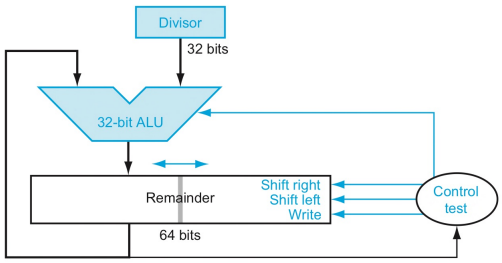


FIGURE 3.11 An improved version of the division hardware. The Divisor register, ALU, and Quotient register are all 32 bits wide, with only the Remainder register left at 64 bits. Compared to Figure 3.8, the ALU and Divisor registers are halved and the remainder is shifted left. This version also combines the Quotient register with the right half of the Remainder register. (As in Figure 3.5, the Remainder register should really be 65 bits to make sure the carry out of the adder is not lost.)

MIPS 指令集中有：

mult $s0, $s1 div $s0, $s1

其中 $s0, $s1 皆為來源暫存器

∵ 目的預設為：H_0 和 L_i 暫存器 (特殊目的 register)

其中：H_0 存乘法結果的前 32 個位元 or 除法結果 remainder

L_i 存乘法結果的後 32 個位元 or = quotient

欲載入 H_i, L_i 的值至一般目的暫存器需使用

mfhi $s0 # $s0 ← H_i

mflo $s1 # $s1 ∈ L_o

然而為何已經有了 mult 指令了，MIPS 還需要 mul $s0, $s1, $s2？

使用 mul 時需注意 $s1 × $s2 結果不可超過 32 bit

∵ 大部份運算不會需要用到太大的 eg. $2^{31} \times 2^{32}$ 的結果

mul 限制的 32 位元內運算已經足夠，又 mul 可使用另外的乘法器

而非 sequential circuit 設計的乘法器，會有需等待 clock 時脈週期的問題

速度可更快。