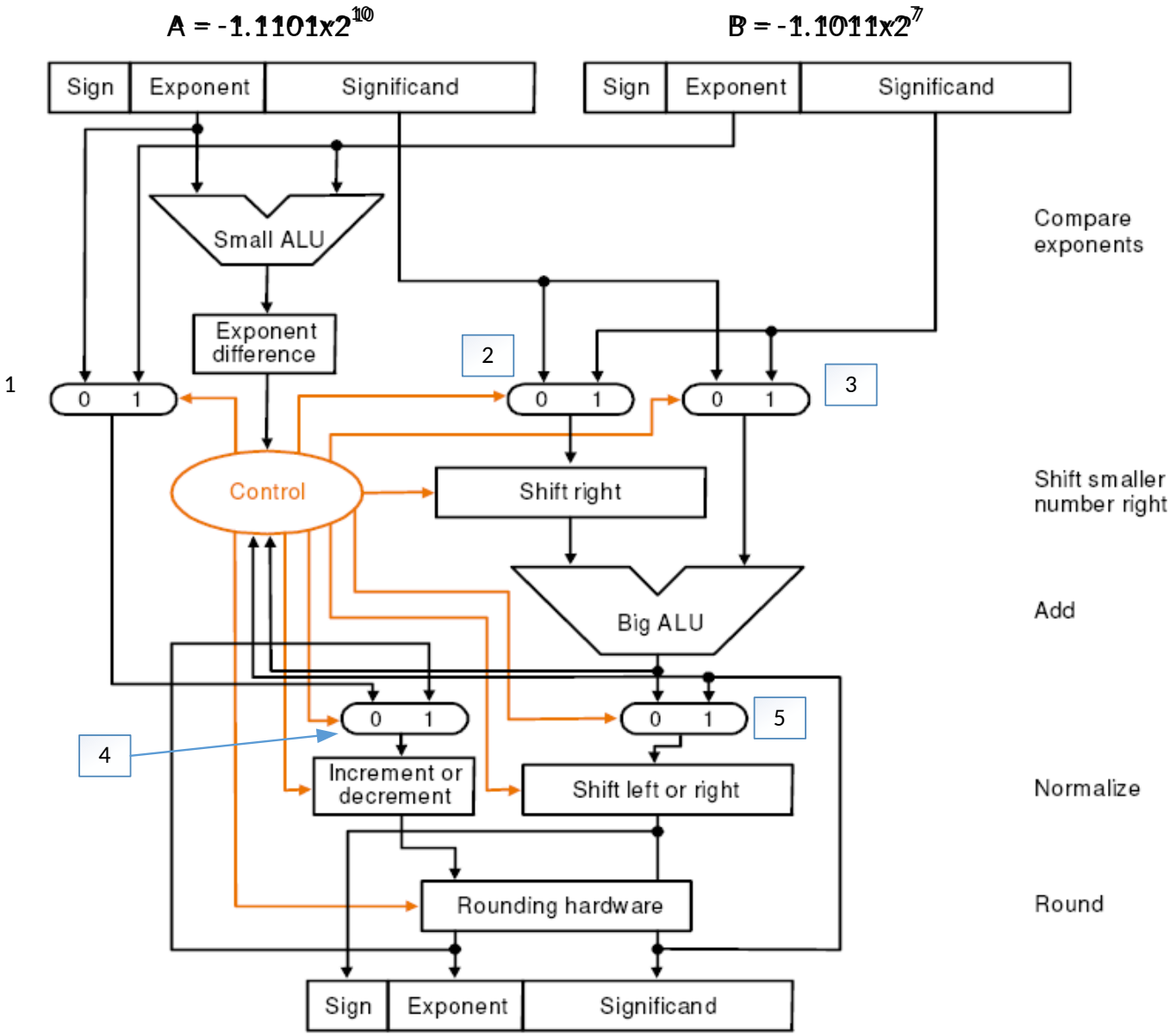
1. (12 points) 浮點數 (IEEE 754) ，參考下圖，執行 -
1.1101×2^{10} - 1.1011×2^{7}

A. 若為單精準度浮點數，計算結果為何 ?(以 16 進制表示)
B. 若為倍精準度浮點數，計算結果為何 ?(以 16 進制表示)
C. 若為單精準度浮點數， Big ALU 及 Small ALU 規格為何？
D. 若為倍精準度浮點數， Big ALU 及 Small ALU 規格為何？
E. Shift left or right 單元在單精準度浮點數計算過程中是否有執行？
F. Increment or decrement 單元在倍精準度浮點數計算過程中是否有執行？


A = -1.1101x2^10
B = -1.1011x2^7
Sign
Exponent
Significand
Sign
Exponent
Significand
Small ALU
Exponent difference
1
0 1
2
0 1
0 1
3
Control
Shift right
Big ALU
0 1
0 1
5
4
Increment or decrement
Shift left or right
Rounding hardware
Sign
Exponent
Significand
Compare exponents
Shift smaller number right
Add
Normalize
Round

2. (10 points) 浮點數 (IEEE 754)

A 單精準度浮點數表示正數的最大值及最小值為何？

B 倍精準度浮點數表示正數的最大值及最小值為何？

C 單精準度浮點數中， $3x10^{-38}$ 是否 underflow? $3x10^{38}$ 是否 overflow?

D. 浮點數的 0 如何表示？ 浮點數的無限大 (infinity) 如何表示？

E. 浮點數指數為何不使用 2 補數表示負數？

3. (10 points) 使用兩種不同方法執行有號數乘法　1101 x 1011 。

4. (6 points) - 280 以 16 bits 表示 , 以 2 補數表示其負數

A. 將此 16 bits 數字移至 8 bits 後，結果為何？

B. 將此 16 bits 數字移至 32bits 後，結果為何？

C. 承 B 小題，若此 32bits 置於記憶體位置 C0h, C1h, C2h, C3h, 說明使用 little-endian 及 big-endian 之差異。

5. (10 points)
   A. 簡扼說明積體電路製程之步驟。
   B. 為何 MIPS 不適合做為計算機效能評估依據？

6. (10 points) 有三個不同的處理器 P1 、 P2 、 P3 可執行相同的指令集。 P1 的時脈速率是 3GHz ， CPI 是 1.6 ， P2 的時脈速率是 2.5GHz ， CPI 是 1.2 ， P3 的時脈速率是 4GHz ， CPI 是 2.5 。

A. P1 、 P2 、 P3 之 MIPS 各為若干？

B. 執行 480,000 個指令， P1 、 P2 、 P3 各需多少時間？

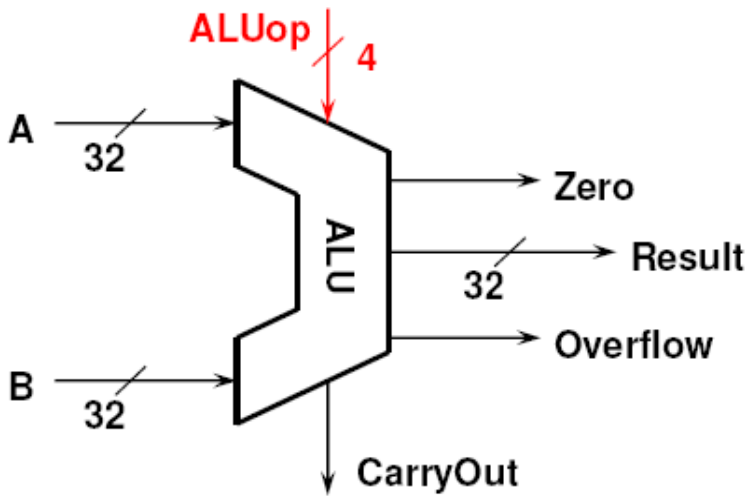
7. (10 points) A 處理器為 Memory-Memory ISA ，有 50 個指令，每個指令皆有 10 種定址法，可定址之記憶體空間為 1M words(一個 word 為 32bit) ，每個指令皆有兩個來源運算元及一個目的運算元 : 指令格式為（**OPcode, Addressing mode, DEST, SRC1, SRC2**) 。
B 處理器為 Accumulator Machine ISA ，有 50 個指令，每個指令皆有 10 種定址法，可定址之記憶體空間為 1M words(一個 word 為 32bit) ，每個指令只有一個目的運算元 : 指令格式為（**OPcode, Addressing mode, Operand**) 。

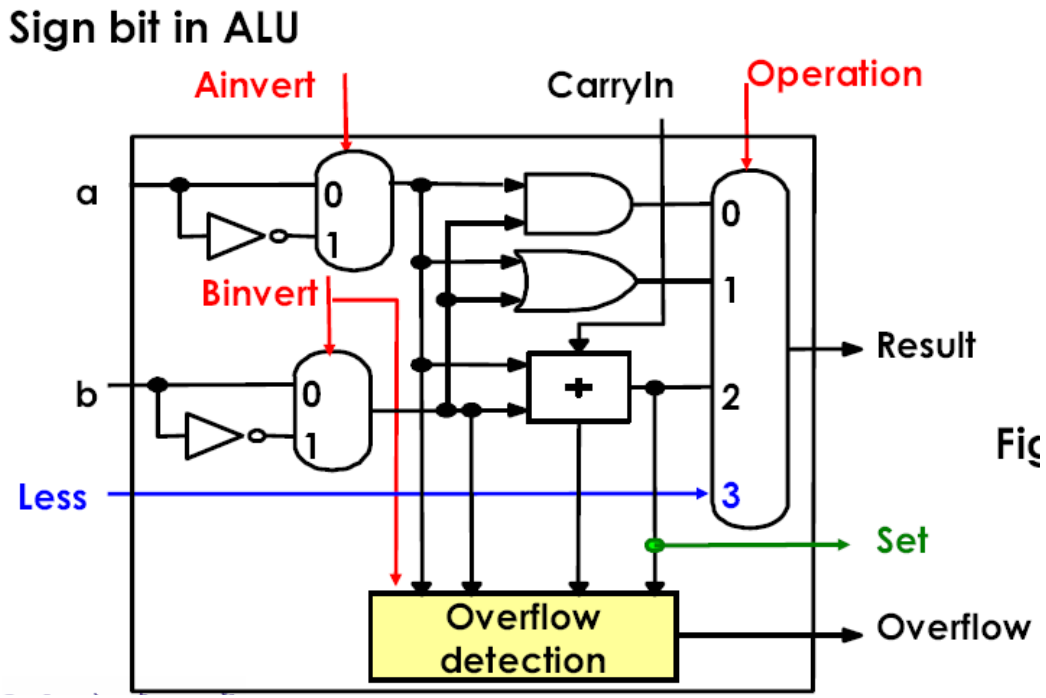
A. A 處理器最小指令格式長度為多少 bit?
B. B 處理器的最小指令格式長度為多少 bit?
C. Memory-Memory ISA 有何缺點？
D. Accumulator Machine 仍存在那些缺點？
E. ISA 為何要有多種定址法？

8. (10 points) 參考下圖：

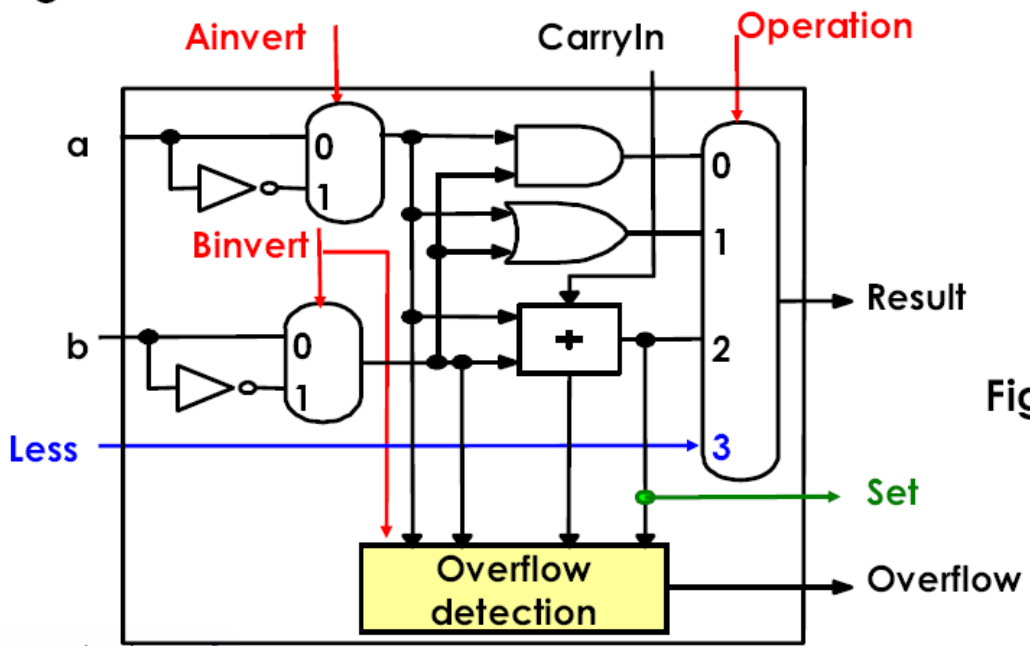
A. 設計 Overflow detection 電路。

B. 設計 (繪出) 一 8 bits ALU, 具有 slt(set on less than) 、零偵測功能及溢位，並以功能表方式表示 ALUop 與運算的關係。 ALUop 為 4 條線 (Ainvert, Binvert, Operation) 。





Sign bit in ALU



Fig

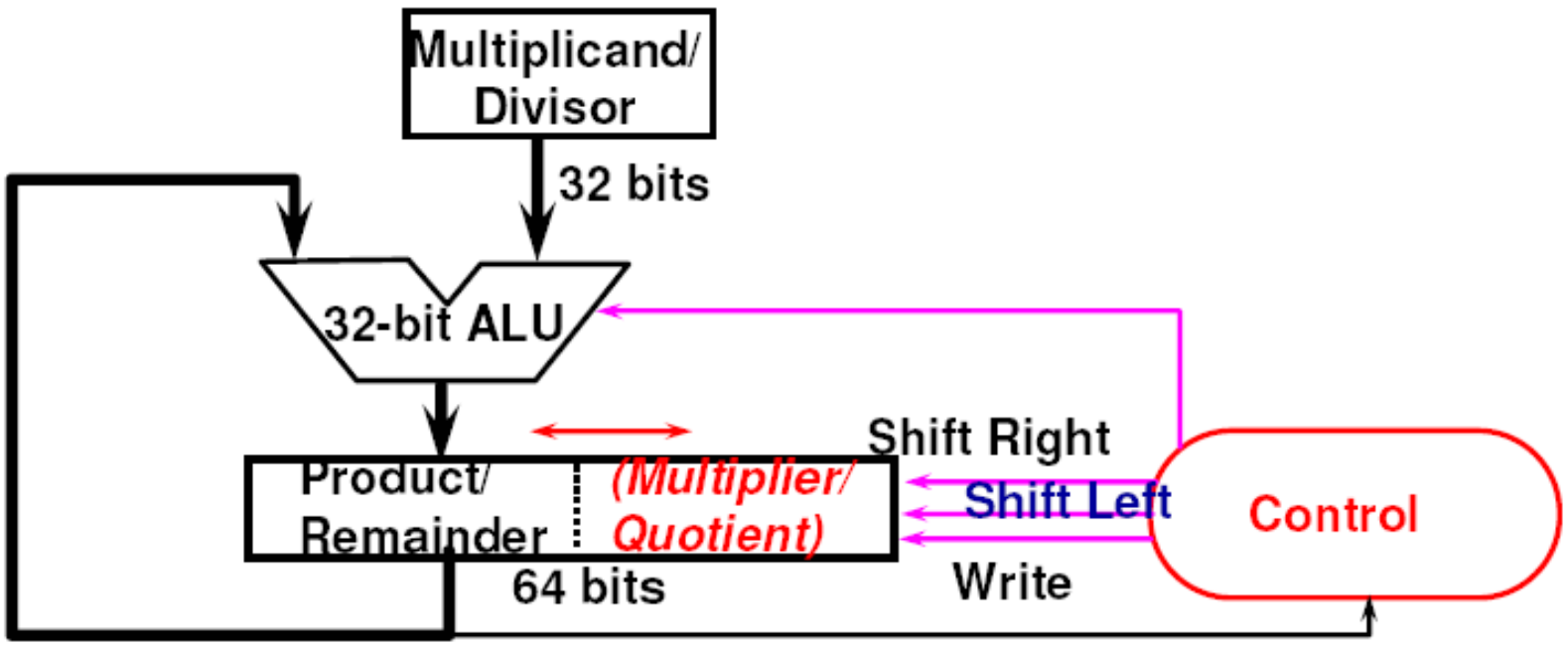
9. (10 points) 參考下圖所附之硬體方塊及流程圖計算

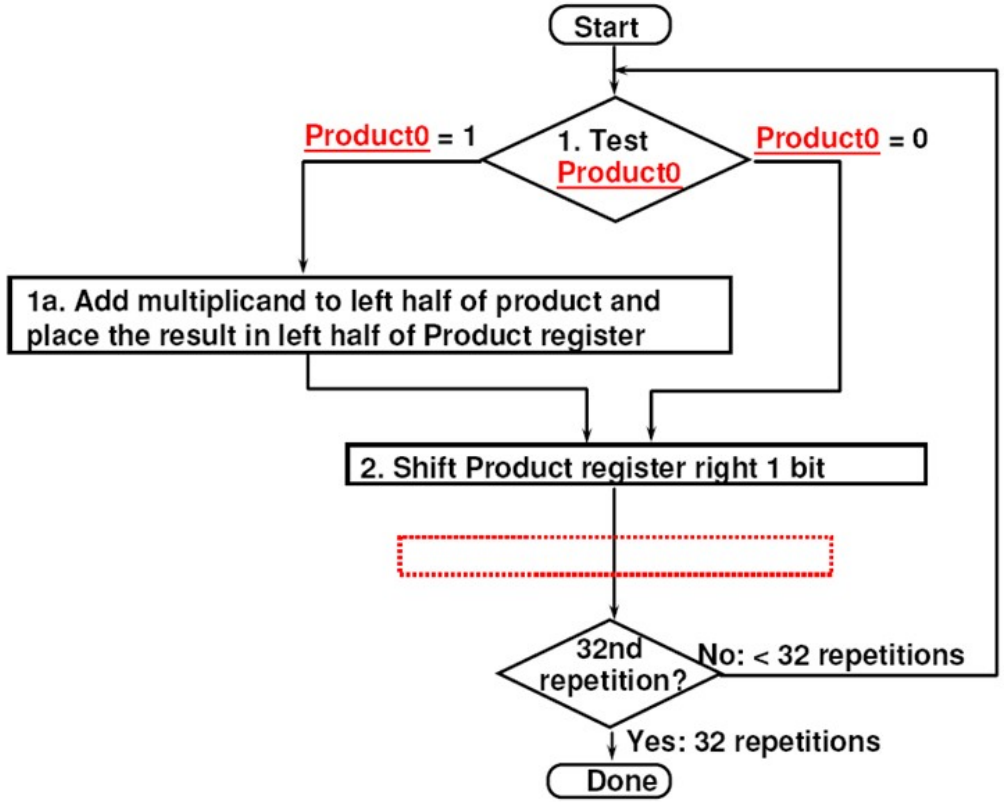
A. 1101 X 1101

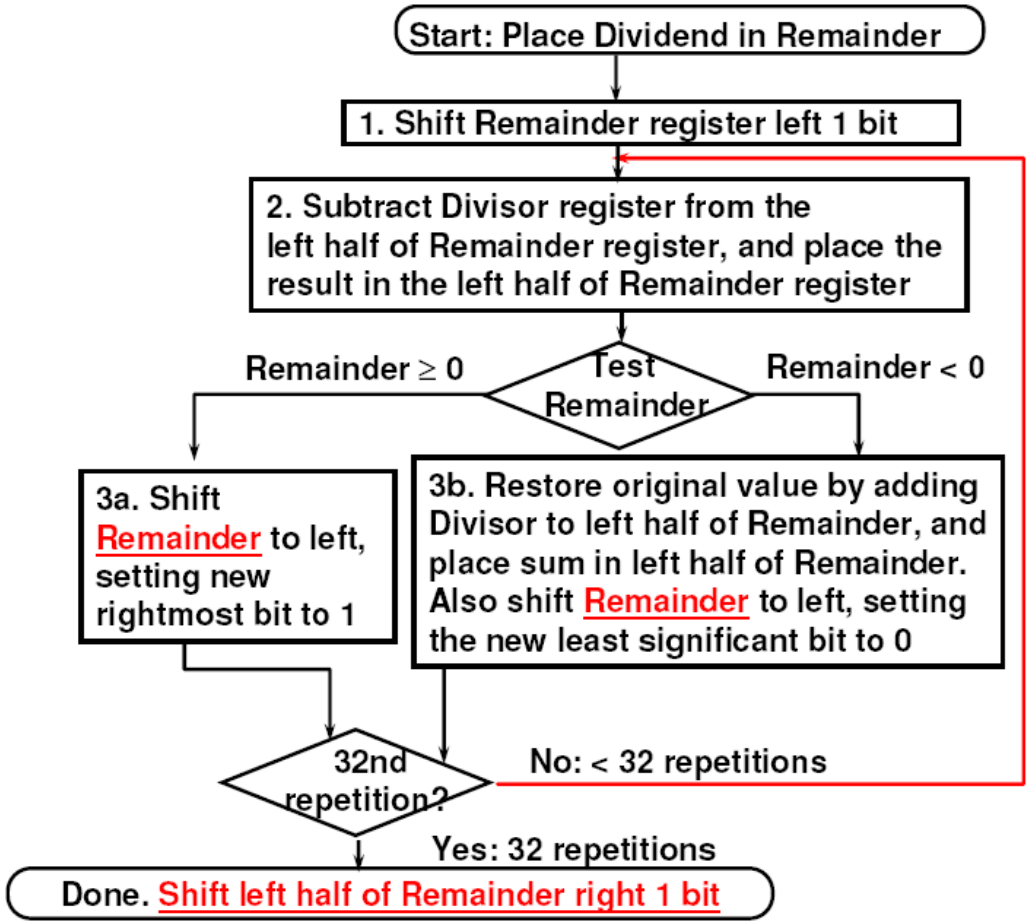
B. 00110101 / 0110

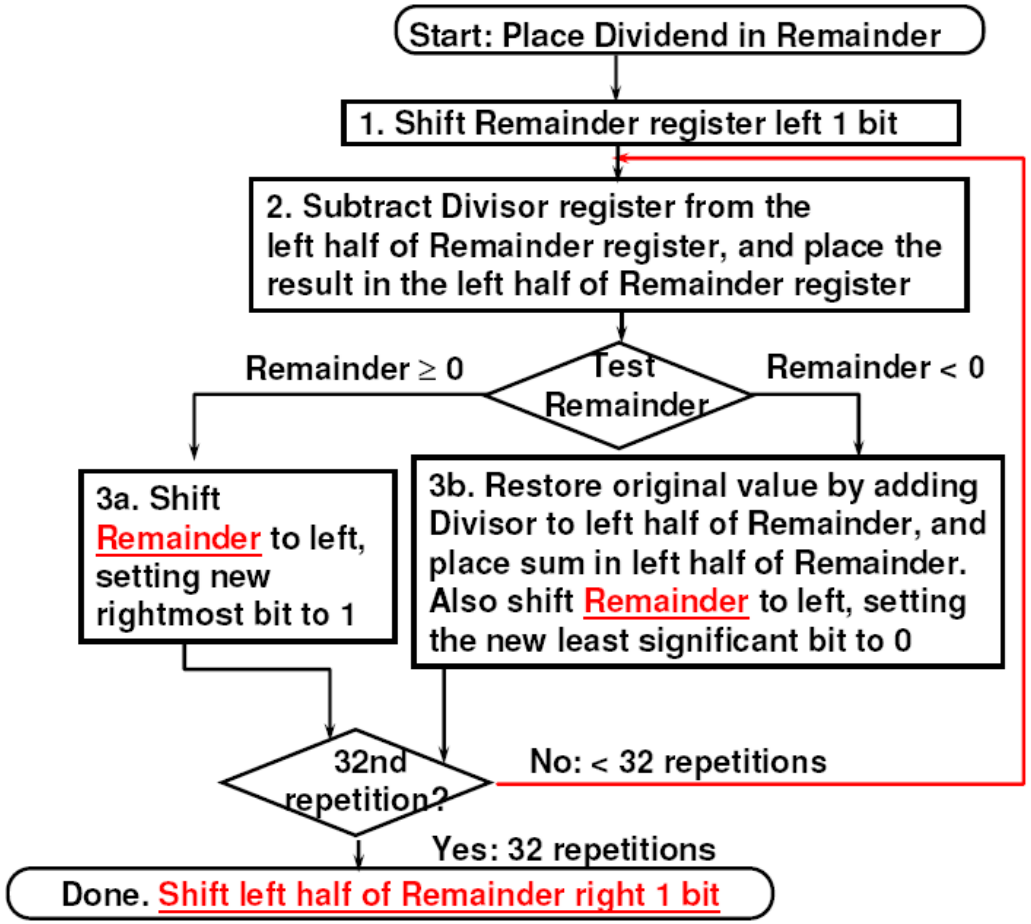
C. 01001000 / 1010

列出每個步驟暫存器之內容與最後結果。




Start
Product0 = 1
1. Test Product0
Product0 = 0
1a. Add multiplicand to left half of product and place the result in left half of Product register
2. Shift Product register right 1 bit
32nd repetition?
No: < 32 repetitions
Yes: 32 repetitions
Done


Start: Place Dividend in Remainder
1. Shift Remainder register left 1 bit
2. Subtract Divisor register from the left half of Remainder register, and place the result in the left half of Remainder register
Test Remainder
Remainder ≥ 0
Remainder < 0
3a. Shift Remainder to left, setting new rightmost bit to 1
3b. Restore original value by adding Divisor to left half of Remainder, and place sum in left half of Remainder. Also shift Remainder to left, setting the new least significant bit to 0
32nd repetition?
No: < 32 repetitions
Yes: 32 repetitions
Done. Shift left half of Remainder right 1 bit


Start: Place Dividend in Remainder
1. Shift Remainder register left 1 bit
2. Subtract Divisor register from the left half of Remainder register, and place the result in the left half of Remainder register
Test Remainder
Remainder ≥ 0
Remainder < 0
3a. Shift Remainder to left, setting new rightmost bit to 1
3b. Restore original value by adding Divisor to left half of Remainder, and place sum in left half of Remainder. Also shift Remainder to left, setting the new least significant bit to 0
32nd repetition?
No: < 32 repetitions
Yes: 32 repetitions
Done. Shift left half of Remainder right 1 bit

10. (8 points) 以 MISP 指令 slt ,bne,beq 實現：

A. ble ： if $s1 <= $s2 then branch

B. bge ： if $s1 >= $s2 then branch

C. blt ： if $s1 < $s2 then branch

D. bgt ： if $s1 > $s2 then branch

條件式分支	若等於則分支	beq $s1,$s2,25	若($s1==$s2)則前往 PC+4+100	等於測試：PC 相對的分支
	若不等於則分支	bne $s1,$s2,25	若($s1!=$s2)則前往 PC+4+100	不等於測試：PC 相對的分支
	若小於則設定	slt $s1,$s2,$s3	若($s2<$s3) $s1=1; 否則 $s1=0	小於比較；用於 beq、bne
	無號若小於則設定	sltu $s1,$s2,20	若($s2<$s3) $s1=1; 否則 $s1=0	無號數的小於比較
	若小於立即值則設定	slti $s1,$s2,20	若($s2<20) $s1=1; 否則 $s1=0	小於某常數的比較
	無號若小於立即值則設定	sltiu $s1,S2,20	若($s2<20) $s1=1; 否則 $s1=0	無號數的小於某常數的比較
非條件式跳躍	跳躍	j 2500	前往 10000	跳至目的位址
	透過暫存器跳躍	jr $ra	前往 $ra	用於 switch 敘述、程序反回
	跳躍並連結	jal 2500	$ra=PC+4 前往 10000	用於程序呼叫

MIPS 組合語言

分類	指　令	舉　例	意　義	註　解
資料傳輸	載入字組	lw $s1,20($s2)	$s1=**Memory**[$s2+**20**]	字組由記憶體載入至暫存器
	儲存字組	sw $s1,20($s2)	**Memory**[$s2+**20**]=$S1	字組由暫存器儲存至記意體
	載入半字組	lh $s1,20($s2)	$s1=**Memory**[$s2+**20**]	半字組由記憶體載入至暫存器
	載入無號半字組	lhu $s1,20($s2)	$s1=**Memory**[$s2+**20**]	無號的半字組由記憶體載入至暫存器
	儲存半字組	sh $s1,20($s2)	**Memory**[$s2+**20**]=$s1	半字組由暫存器儲存至記憶體
	載入位元組	lb $s1,20($s2)	$s1=**Memory**[$s2+**20**]	位元組由記憶體載入至暫存器
	載入無號位元組	lbu $s1,20($s2)	$s1=**Memory**[$s2+**20**]	無號的位元組由記憶體載入至暫存器
	儲存位元組	sb $s1,20($s2)	**Memory**[$s2+**20**]=$s1	位元組由暫存器儲存至記憶體
	載入連結的字元組	ll $s1,20($s2)	$s1=**Memory**[$s2+**20**]	作為不可分割的（記憶體與儲存器內容）交換中第一部份的載入字元組
	條件式儲存字元組	sc $s1,20($s2)	**Memory**[$s2+**20**]=$s1; $s1=0 或 1	作為不可分割的（記憶體與暫存器內容）交換中第二部份的儲存字組
	載入上半部立即值	lui $s1,20	$s1=**20***$2^{16}$	載入常數至較高的 16 位元
邏輯	及	and $s1,$s2,$s3	$s1=$s2 **&** $s3	三個暫存器運算元：逐位元的及運算
	或	or $s1,$s2,$s3	$s1=$s2|$s3	三個暫存器運算元：逐位元的或運算
	反或	nor $s1,$s2,$s3	$s1=～($s2|$s3)	三個暫存器運算元：逐位元的反或運算
	及立即值	andi $s1,$s2,20	$s1=$s2 **& 20**	暫存器與常數做逐位元的及運算
	或立即值	ori $s1,$s2,20	$s1=$s2|**20**	暫存器與常數做逐位元的或運算
	邏輯左移	sll $s1,$s2,10	$s1=$s2<<**10**	左移常數個位元位置
	邏輯右移	srl $s1,$s2,10	$s1=$s2>>**10**	右移常數個位元位置

MIPS 組合語言

分類	指　令	舉　例	意　義	註　解
條件式分支	若等於則分支	beq $s1,$s2,25	**若**($s1==$s2)則**前往 PC+4+100**	等於測試：PC 相對的分支
	若不等於則分支	bne $s1,$s2,25	**若**($s1!=$s2)則**前往 PC+4+100**	不等於測試：PC 相對的分支
	若小於則設定	slt $s1,$s2,$s3	**若**($s2<$s3) $s1=**1**; **否則** $s1=**0**	小於比較；用於 beq、bne
	無號若小於則設定	sltu $s1,$s2,20	**若**($s2<$s3) $s1=**1**; **否則** $s1=**0**	無號數的小於比較
	若小於立即值則設定	slti $s1,$s2,20	**若**($s2<**20**) $s1=**1**; **否則** $s1=**0**	小於某常數的比較
	無號若小於立即值則設定	sltiu $s1,S2,20	**若**($s2<**20**) $s1=**1**; **否則** $s1=**0**	無號數的小於某常數的比較
非條件式跳躍	跳躍	j　2500	**前往 10000**	跳至目的位址
	透過暫存器跳躍	jr　$ra	**前往** $ra	用於 switch 敘述、程序反回
	跳躍並連結	jal 2500	$ra=**PC+4 前往 10000**	用於程序呼叫

11. (15 points) 將下列 C code 翻成 MIPS 指令

A. if (i <= 10) f = g*2;
else f = g/2; //f, g, h,i, j in $s0, $s1, $s2, $s3, $s4

```
    slti $t0, $s3, 11    //if $s3 < 11, $t0 = 1
    beq  $t0, $0, ELSE   //if $t0 = 0, f = gx2
    Sll  $s0, $s1, 1
    j   EXIT
ELSE:  Srl  $s0, $s1, 1
EXIT:
```

B. while (D[i] >= k+1) i = i-1; //i in $s3, k in $s5, address of D in $s6

```
Loop: sll  $t1, $s3, 2
      add  $t1, $t1, $s6
      lw   $t0, 0($t1)
      addi $t2, $s5, 1
      slt  $t3, $t0, $t2
      bne  $t3, $0, Exit
      addi $s3, $s3, -1
      j    Loop
Exit: …
```

C. A = C[4] << 4 //A in $t1, address of C in $s1

```
Lw  $t0, 16($S1)
Sll $t1, $t0, 4
```

D. A = C[4] << i //I in $s2, A in $t1, address of C in $s1

```
       Lw  $t0, 16($S1)        //$t0 = c[4]
       add $t2, $s2, $0        // $t2 = i
Loop:  Beq $t2, $0 , EXIT      // if i = 0, jump exit
       Sll $t0, $t0, 1         //$t0 = $t0 << 1
       Addi $t2,$t2, -1        // $t2 = $t2  - 1
    j loop
EXIT:  add $t1, $t0, $0        // A = c[4] << i
```

E. for (i = a; i> 0; i--)
for (j = b; j> 0; j--)
D[i+j] = D[i+j] + i +j; //i, j, a, b in $s1, $s2, $s3, $s4, address of D in $s5

```
    add $s1,$s3, $0     // i = a
    beq $0, $0 TEST1    //jump test1
LOOP1 :add $s2, $s4, $0    // j = b
    beq $0, $0, TEST2  // jump test2
LOOP2 :add $t1, $s1, $s2  // $t1 = i+j
    sll $t2, $t1, 2    // $t2 = $t1 x 4
    add $t2, $t2,$s5   // $t2 = &D[i+j]
    lw  $t3, 0($t2)    // $t3 = D[i+j]
    add $t3, $t3, $t1  //$t3 = D[i+j] +i+j
    sw  $t3, 0($t2)    // D[i+j] = D[i+j] +i+j
    addi $s2, $s2, -1  // j--
TEST2 : bne $s2, $0, LOOP2 // test j = 0? Jump loop2 if j != 0
    addi $s1, $s1, -1  // i--
TEST1 : bne $S1, $0, LOOP1 // test i = 0? Jump loop1 if i != 0
```