公式推導

vver = uver – g\*t => t = - (vver - uver) / g

x = vhor \* (2\*t)

= -2 \* vhor \* (vver - uver) / g

= -2 \* (16\*cos θ) \* (-16\*sin θ) / 10

= 16\*16\*sin(2\*θ) / 10

= 256\*sin(2\*θ) / 10

因此θ = sin-1(10\*x / 256) / 2

Table 建立

0 <= θ <= 45

0 <= 2\*θ <= 90

所以需要建立sin00至sin900 的table

Table中記錄的數值是是sin值乘上256的結果（因為一個register有8bits，而所需sin值會介於0-1之間，因此乘上256能最充分使用8bits的空間）

這裡以間距10來建立table（因此有可能與正確答案相差0.50）

sin table的數值是用c++來計算後再貼上程式碼的（getTable.cpp）

最後由於一個register最大只能儲存到255，因此當sin值很接近1時，該值乘上256後為256， 但無法被儲存，這裡存成255

演算法過程

Input ：假設題目要計算距離為x的最佳角度，則在input(0x001)的位置放上10\*x (以整數來看待)

這時公式可以看成 10\*x = 256\*sin(2\*θ), 而右式為我們所存的sin table值

1. 把sin table的值依序從00 到 900 取出，並與input(也就是10\*x) 比較，直到遇到比input 大的才停止
2. 將此距離GREATER\_DIS（略大於10\*x的距離）與比它小0.50之步級角所能到達的距離PRE\_DIS存起來（此時PRE\_DIS <= 10\*x < GREATER\_DIS）
3. 若10\*x 較接近PRE\_DIS, 則取能達到PRE\_DIS之步級角， 反之則取能達到GREATER\_DIS之步級角。若10\*x剛好介於兩者之間，則取PRE\_DIS，（這樣會得到更準確的值，因為考慮x對θ的圖，可知在題意範圍內任取兩點作一直線，此直線除了末端兩點之外必定在曲線之下，因此可知PRE\_DIS之步級角會更接近實際角度）

由於建立table時最大為255，因此若輸入為25.5，則找到最後也不會找到比255大的值，因此程式裡是將這個輸入做另外處理(寫在Table的最後部分的special case， 做法是將該題的答案輸出到0x02後結束)

如何優化

精準度：用2個register來儲存sin table的值，這樣我們更準確的判斷到底哪個步級角會發射到的距離更接近我們所想要達到的距離，但這會增加程式碼記憶體空間。

程式碼記憶體空間減小：比如將sin table的間距改為20， 再用一樣的方法解，只是這樣步級角就只有整數了，但我們還可以在當input在PRE\_DIS 與 GREATER\_DIS 之間且與兩者的距離相差不大時，將答案設成兩者之步級角之平均，那麼就一樣可以得到有0.5度之步級角了，只是這樣還是會降低精準度。

演算法：目前尋找sin table中最靠近input的值是用for loop的概念去找，因此時間複雜度為O(n)， 若改為binary search， 則可將時間複雜度降低成O(log n)