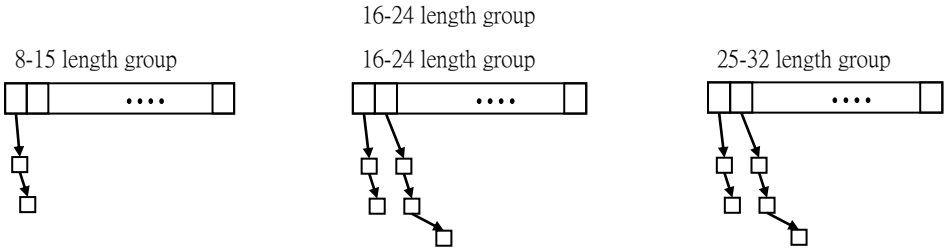
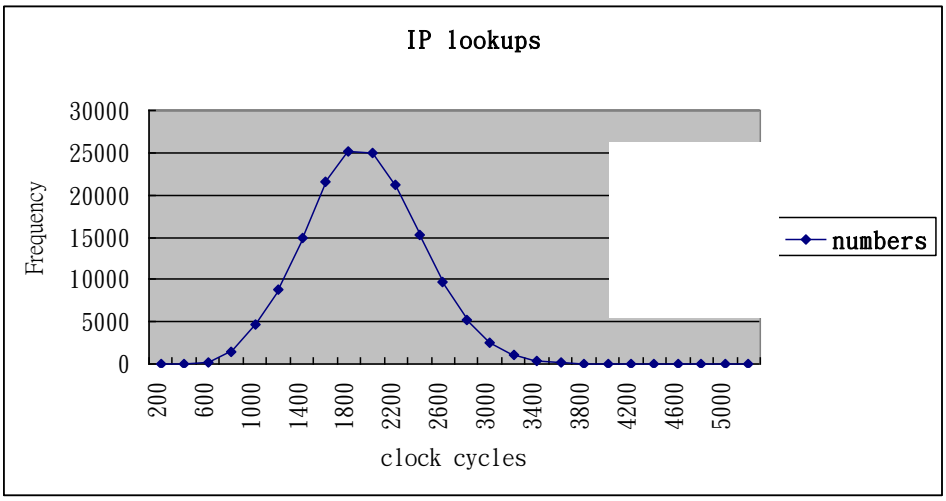
C Language Programming: Homework #8
Assigned on 12/29/2015(Tuesday), Due on 01/12/2016(Tuesday)

1. 請寫一個程式去建立 IP table (ip_table)裡的資料，並且對所建立好的資料結構分別紀錄 search，update 及 delete 這三個動作的平均時間。
2. 建立方式如下：
   Divide prefixes of lengths 8-15, 16-24, and 25-32 into three length groups and store in 3 arrays called segmentation tables of sizes 2^8, 2^{12}, and 2^{12}. Use the first 8 or 12 significant bits of the prefixes for the segmentation table in each length group.
   IP Prefix Format：IP / Length (e.g.：4.0.0.0 4.17.255.0/24)



   在上述 ip_table 中，0.0.0.0/8 and 0.2.64.0/15 這筆資料因其 prefix length=8，需插入於 8≦prefix length <16 這個 Array 中的 **0^{th} element** 這個位置； 0.3.0.0/23、0.3.8.0/22、0.19.33.0/24、0.19.38.0/23 及 0.20.41.0/24 這五筆資料 prefix length 皆大於 16，因此需分別插入於 16≦prefix length≦24 這個 Array 中的 **0^{th} element and 1^{th} element** 這兩個位置，
3. Use the prefixes in the given prefix file to create the above data structure.
4. Use the ip in the given search file 丟入所建立好的資料結構去做 search (longest prefix match)，並請紀錄 search 所需要的時間。
5. For insertion, use the given insert file to 插入原先已建立好的資料結構，並請紀錄插入所需要的時間。
6. For deletion，use the given delete file to 對原先建立好的資料結構作 delete 的動作，並請紀錄 delete 所需要的時間。
7. 報告需求：
   數據需包含 average search clock cycles，average insert clock cycles，average delete clock cycles，並請紀錄個別的 clock cycles 製作統計圖表如下。



e.g. 進行 search 需要 1400 clock cycles 的次數有 15000 次

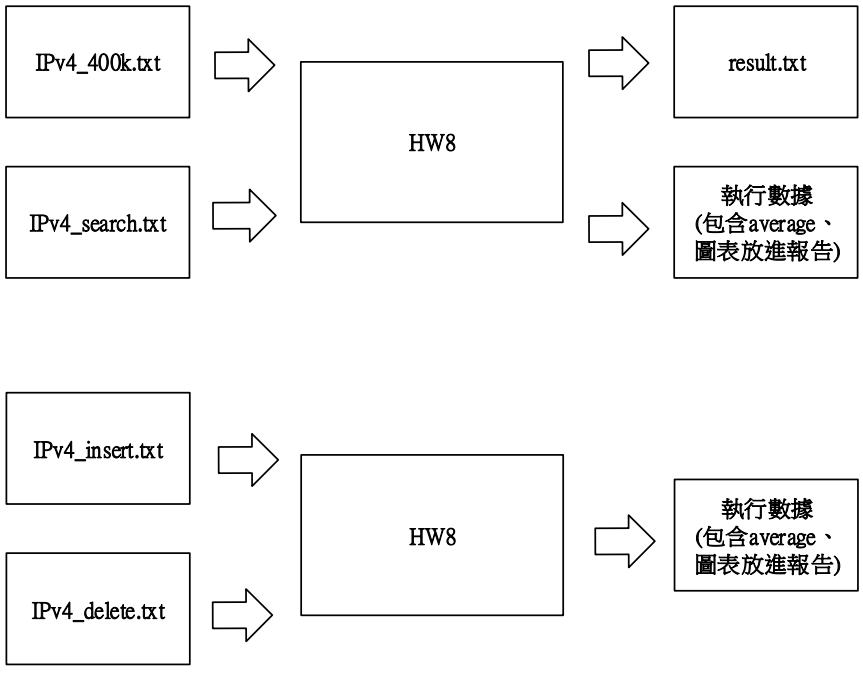
8. Score

(1) create and search	60%
(2) insert	20%
(3) delete	20%

9. File

Input file at /home/data/hw8	
IPv4_400k.txt	此檔案內容為 prefix，用來 create 出資料結構 ※第一條 prefix 0.0.0.0/0 代表 default prefix， 任何 ip 都會 match (0 bit 的比較)
IPv4_search.txt	此檔案內容為 ip，用來進行 search
IPv4_insert.txt	此檔案內容為 prefix，用來進行 insert
IPv4_delete.txt	此檔案內容為 prefix，用來進行 delete

Output file	
result.txt	此檔案為進行完 search 產生的結果。
result format	140.116.0.0/16 24.112.34.0/24 … (印出結果為 match 的 prefix，請參考 appendix 按照 search 檔案內的順序，請勿增加說明)



※insert 跟 delect 完不用產生 result 檔案

10. rdtsc: count the number of clock cycles of CPU

Function	How to use?

inline unsigned long long int rdtsc()
{
  unsigned long long int x;
  asm volatile ("rdtsc" : "=A" (x));
  return x;
}
	unsigned long long int begin,end, result= 0;
	begin=rdtsc();
	// executing code you want to measure
	end=rdtsc();
	result = end-begin;

11. Appendix

LPM (Longest Prefix Match):

假設 create 完之後有兩條資料 101.64.0.0/10、101.104.0.0/16，

舉 search ip 101.104.1.0 而言，

因為第一條資料 01100101.01000000.00000000.00000000 及

第二條資料 01100101.01101000.00000000.00000000，

兩條資料都會 match，但因為 LPM 的特性，match 結果為第二條。