Zadanie polega na napisaniu programu umożliwiającego dokładne obliczenie czasu wykonania funkcji przez procesor ARM. Do tego celu należy odczytać stan odpowiedniego rejestru procesora odpowiadającego ze stan licznika (tzw. performance counter).

Część A)

1. Napisać program tworzący tablicę o rozmiarze SIZE (gdzie SIZE jest określone przez #define). Na początku programu wypełnić tablicę kolejnymi liczbami naturalnymi, ale w odwrotnej kolejności (zaczynając od końca tablicy).
2. Znaleźć w internecie funkcję bubblesort do sortowania bąbelkowego i użyć jej w programie do posortowania elementów tablicy.
3. Na podstawie wątku (w szczególności na podstawie najczęściej plusowanej odpowiedzi w tym wątku):

   https://stackoverflow.com/questions/3247373/how-to-measure-program-execution-time-in-arm-cortex-a8-processor/3250835

   napisać funkcje inicjalizujące licznik procesora oraz odczytujące stan licznika procesora. Odczytać licznik przed i po funkcji bubblesort w celu obliczenia liczby cykli, które upłynęły podczas sortowania.

4. Skompilować i uruchomić program. Wyjaśnić dlaczego program nie wykonuje się poprawnie (błąd „Illegal instruction").

Część B)

1. Na podstawie wyżej wspomnianego wątku (należy dokładnie przeczytać wszystkie kolejne odpowiedzi na najczęściej plusowaną odpowiedź) napisać krótki program włączający dostęp do liczników procesora ARM z poziomu użytkownika.
2. Na podstawie wątku:

   https://stackoverflow.com/questions/3467850/cross-compiling-a-kernel-module

   napisać odpowiedni Makefile do kompilowania tego programu tak, aby powstał z niego tzw. moduł jądra (kernel module, plik z rozszerzeniem .ko). Uwaga1: Należy pamiętać, że powinna to być kompilacja skrośna (cross-compilation). Uwaga2: program należy kompilować w oddzielnym folderze.

3. Skopiować plik .ko na płytkę Devkit. Wstrzyknąć moduł do kernela za pomocą polecenia insmod. Sprawdzić, czy moduł zainicjalizował się poprawnie za pomocą polecenia dmesg.
4. Ponownie uruchomić pierwszy program (z części A). Zmieniając rozmiar tablicy (SIZE), obliczyć liczbę cykli potrzebnych do posortowania tablicy o 10, 100, 1000, 10000 i 100000 elementach. Określić zależność między rozmiarem tablicy a liczbą cykli potrzebnych do jej posortowania metodą bąbelkową.
5. Zmienić poziom optymalizacji dla kompilatora gcc na O3. Powtórzyć pomiary z punktu 4. Jak zmieniła się szybkość sortowania?