DCL

Napisać w asemblerze program symulujący działanie maszyny szyfrującej DCL. Maszyna DCL działa na zbiorze *dopuszczalnych znaków* zawierającym: duże litery alfabetu angielskiego, cyfry 1 do 9, dwukropek, średnik, pytajnik, znak równości, znak mniejszości, znak większości, małpę. Jedynie znaki z tego zbioru mogą się pojawić w poprawnych parametrach programu oraz w poprawnym wejściu i wyjściu programu.

Maszyna składa się z trzech bębenków szyfrujących: lewego L, prawego R i odwracającego T. Bębenki L i R mogą się obracać i każdy z nich może znajdować się w jednej z 42 pozycji oznaczanych znakami z dopuszczalnego zbioru. Maszyna zamienia tekst wejściowy na wyjściowy, wykonując dla każdego znaku ciąg permutacji. Jeśli bębenek L jest w pozycji l, a bębenek R w pozycji r, to maszyna wykonuje permutację

$$Q_r^{-1}R^{-1}Q_r\, Q_l^{-1}L^{-1}Q_l\, T\, Q_l^{-1}LQ_l\, Q_r^{-1}RQ_r$$

gdzie L, R i T są permutacjami bębenków zadanymi przez parametry programu. Procesy szyfrowania i deszyfrowania są ze sobą zamienne.

Permutacje Q dokonują cyklicznego przesunięcia znaków zgodnie z ich kodami ASCII. Przykładowo Q_5 zamienia `1` na `5`, `2` na `6`, `9` na `=`, `=` na `A`, `A` na `E`, `B` na `F`, `Z` na `4`, a $Q_=$ zamienia `1` na `=`, `2` na `>`, `?` na `K`. Permutacja Q_1 jest identycznością. Permutacja T jest złożeniem 21 rozłącznych cykli dwuelementowych (złożenie TT jest identycznością). X^{-1} oznacza permutację odwrotną do permutacji X. Złożenie permutacji wykonuje się od prawej do lewej.

Przed zaszyfrowaniem każdego znaku bębenek R obraca się o jedną pozycję (cyklicznie zgodnie z kodami ASCII pozycji), czyli jego pozycja zmienia się na przykład z `1` na `2`, z `?` na `@`, z `A` na `B`, z `B` na `C`, z `Z` na `1`. Jeśli bębenek R osiągnie tzw. pozycję obrotową, to również bębenek L obraca się o jedną pozycję. Pozycje obrotowe to `L`, `R`, `T`.

Kluczem szyfrowania jest para znaków oznaczająca początkowe pozycje bębenków L i R.

Program przyjmuje cztery parametry: permutację L, permutację R, permutację T, klucz szyfrowania. Program czyta szyfrowany lub deszyfrowany tekst ze standardowego wejścia, a wynik zapisuje na standardowe wyjście. Po przetworzeniu całego wejścia program kończy się kodem 0. Program powinien sprawdzać poprawność parametrów i danych wejściowych, a po wykryciu błędu powinien natychmiast zakończyć się kodem 1. Czytanie i zapisywanie powinno odbywać się w blokach, a nie znak po znaku.

Oceniane będą poprawność i szybkość działania programu, zajętość pamięci (rozmiary poszczególnych sekcji), styl kodowania. Tradycyjny styl programowania w asemblerze polega na rozpoczynaniu etykiet od pierwszej kolumny, mnemoników od dziewiątej kolumny, a listy argumentów od siedemnastej kolumny. Inny akceptowalny styl prezentowany jest w przykładach pokazywanych na zajęciach. Kod powinien być dobrze skomentowany, co oznacza między innymi, że każda procedura powinna być opatrzona informacją, co robi, jak przekazywane są do niej parametry, jak przekazywany jest jej wynik, jakie rejestry modyfikuje. To samo dotyczy makr. Komentarza wymagają także wszystkie kluczowe lub nietrywialne linie wewnątrz procedur lub makr. W przypadku asemblera nie jest przesadą komentowania prawie każdej linii kodu, ale należy jak ognia unikać komentarzy typu „zwiększenie wartości rejestru rax o 1”.

Dołączone do zadania przykłady składają się z trójek plików. Plik `*.key` zawiera parametry wywołania programu, a pliki `*.a` i `*.b` zawierają parę tekstów odpowiadających sobie przy szyfrowaniu i deszyfrowaniu.

Jako rozwiązanie należy oddać plik `dcl.asm`. Program będzie kompilowany poleceniami:

```
nasm -f elf64 -w+all -w+error -o dcl.o dcl.asm
ld --fatal-warnings -o dcl dcl.o
```