Zadanie 2 – zajęcia 3 i 4

Operacje na plikach; praca z napisami.

Cel ćwiczenia

Celem zadani realizowanego w trakcie trzeciego i czwartego laboratorium jest zaznajomienie studentów z funkcjami przerwania 21h realizującymi operacje wejścia/wyjścia na plikach oraz z przetwarzaniem napisów w asemblerze.

Zadanie

W ramach zadania 2 studenci zaimplementują program wczytujący zawartość pliku wejściowego, dokonujący na niej określonych operacji (szczegóły w kolejnym podpunkcie instrukcji) i zapisujący rezultat do pliku wynikowego. Nazwy plików wejściowego i wyjściowego, jak również ewentualne inne parametry, są przekazywane w lini poleceń programu.

Implementacja rozwiązania zadania 2 powinna obejmować weryfikację poprawności parametrów przekazanych w linii poleceń. Pliki wskazane w parametrach należy otwierać odpowiednio w trybie do odczytu (plik wejściowy) lub do zapisu (plik wynikowy). Należy również uwzględnić obsługę błędów, które mogą zostać zgłoszone przez przerwanie 21h w trakcie realizacji operacji na plikach.

W pełni prawidłowe rozwiązanie zadania powinno umożliwiać pracę z plikami o dowolnym rozmiarze. Proszę jednak zwrócić uwagę, że wczytywanie / zapisywanie plików znak po znaku jest niewydaje. Z tego powodu pliki powinny być wczytywane, przetwarzane oraz (jeśli to konieczne) zapisywane porcjami, np. Po 16 kilobajtów. Można w tym celu zaimplementować proste funkcje buforujące, np. getchar / putchar. Funkcja zwracająca znak (getchar) powinna wczytać partię pliku do bufora w pamięci, a następnie zwracać poszczególne znaki z tego bufora. Po wyczerpaniu znaków w buforze funkcja powinna wczytać kolejną partię pliku. W analogiczny sposób można zaimplementować funkcję zapisującą (putchar) – funkcja ta umieszcza znaku w buforze, który jest fizycznie zapisywany do pliku, gdy zabraknie w nim miejsca na kolejny znak (oraz tuż przed zakończeniem programu).

Na ocenę rozwiązania zadania składają się:

1. Poprawność implementacji – z uwzględnieniem wymienionych powyżej elementów składowych rozwiązania zadania.

2. Przejrzystość implementacji, w tym należyte skomentowanie poszczególnych partii instrukcji w programie, unikanie nadmiarowych / nieczytelnych instrukcji skoku oraz prawidłowy podział programu na podprocedury.

3. Terminowość realizacji.

Operacja na treści pliku wejściowego – kompresja RLE

Program powinien sprawdzić, czy linia poleceń ma jedną z dwóch poniższych postaci:

```
nazwa_programu input output
nazwa_programu -d input output
```

gdzie `input` to nazwa pliku wejściowego, zaś `output` to nazwa pliku wyjściowego. W pierwszym przypadku, program powinien dokonać konwersji RLE (*run length encoding)* pliku wejściowego I zapisać wynik w pliku wyjściowym. W drugim przypadku program powinien dokonać dekompresji zawartości pliku wejściowego I zapisać wynik do pliku wyjściowego.

Kodowanie RLE polega na zastąpieniu wielokrotnego (>3) powtórzenia pojedynczego bajtu następującymi trzema bajtami: znakiem modyfikacji (*escape character),* licznikiem powtórzeń I powtarzanym bajtem. Jako znak modyfikacji najwygodniej jest przyjąć znak `0x00`. Przykładowo, 10-cio krotne wystąpienieznaku ASCII `0x51` należy zastąpić następującymi trzema bajtami: `0x00 0x0A 0x51`. Aby zapewnić możliwość wiernej dekompresji danych, każde wystąpienie znaku `0x00` w treści oryginalnego)nie skompresowanego) pliku należy zamienić na parę bajtów `0x00 0x00`. W trakcie dekompresji każdą parę bajtół `0x00 0x00` zamieniamy na znak `0x00`. Ponieważ na licznik powtórzeń rezerwujemy 1 bajt, nie może on przekroczyć wartości 255. Jeśli więc w pliku wejściowym występuje ciąg więcej niż 255 identycznych bajtów to kompresujemy osobno każde 255 bajty tego ciągu.

RLE jest przydatny do szybkiej I bezstratnej kompresji zdjęć o niewielkiej ilości kolorów (zdjęcia monochromatyczne lub w 256 odcieniach szarości). Jednolite obszary takich zdjęć (czarne tło I inne powierzchnie o stałym kolorze) reprezentowane są jako wielokrotne powtórzenie tego samego bajtu.