Algorytm:

1. Przekopiowanie argumentów (dwóch dużych liczb) do pomocniczych buforów
2. Przesunięcie mnożnika o jeden bit w prawo i zapisanie jej najmłodszego bitu do flagi przeniesienia CF (zakładamy, że korzystamy z instrukcji `rcr`).
    a. Jeśli jest przeniesienie (flaga jest ustawiona), to do bufora z sumą częściową (`partial`) dodawana jest mnożna.
    b. Jeśli nie ma przeniesienia, to kontynuuj.
3. Przesunięcie mnożnej o jeden bit w lewo - żeby móc dodać ją jako wynik częściowy.
4. Operacja powtarzana w pętli od punktu drugiego, do drugiego - przydałby się jakiś warunek końcowy, ale do tego trzeba znać długości argumentów, aby ustawić licznik.

Pytania:

1. Wprowadzanie dwóch dużych liczb:
    a. odczyt z pliku w systemie hex lub dziesiętnym + konwersja liczb na odpowiedni system
    b. zapisane w programie jako zmienne
    c. odczyt z klawiatury w hex lub dziesiętnym
    d. funkcję do mnożenia wywołać z kodu w C, a samo pobranie liczb byłoby od użytkownika za pomocą `scanf`, które przy okazji dokona konwersji z systemu dziesiętnego. Drukowanie wyniku przez `printf`.
    Wtedy na argumenty do asemblerowej funkcji mnożenia byśmy przekazywały:
    - adresy buforów/tablic (first, second, result, partial)
    - dwie liczby wczytane do zmiennych (multiplier, multiplicand)
    Problem: przechowywanie liczb 1024b nie byłby możliwy w jednej zmiennej (np. long long int ma 64b)
    e. ograniczenia liczb: jak maksymalnie duże powinny być, czy liczby mogą być różnej długości

2. Czy dyrektywa `.fill` to dobry pomysł?

   Na początku funkcji chcemy wyzerować górne części buforów, w którym będziemy robić rotacje. `.fill`, która wypełnia bufor o zadanej wielkości samymi zerami. Deklaracja:

   ```
   first: .fill 512
   ```

   AT&T Assembly Language, s.96:

   "The .fill directive enables the assembler to automatically create the 10,000 data elements for you. The default is to create one byte per field, and fill it with zeros”

   Ewentualnie `.zeros` robi dokładnie to samo.

   stackoverflow:

   While filling a data section, the .zero directive fills the number of bytes specified by expression with zero (0).

   Wada: taki plik więcej waży; dokładnie więcej o te X bajtów wypełnionych zerami (przykład tego jest w tym AT&T na stronie 96). Dla naszych buforów (np. 512B każdy) program by ważył 2048B + waga instrukcji, samych buforów itp.