**ELiAK**

1. **Przedstaw w standardzie IEEE 754 liczbę w pojedynczej i podwójnej precyzji.**

Pojedyncza precyzja: znak 1 bit, wykładnik 8 bitów, mantysa 23 bity.

Obliczenia:

1. = ;
2. przesunięcie przecinka o jedno miejsce w prawo (-1). Liczba wygląda teraz następująco . Aby obliczyć wykładnik należy zsumować -1 oraz 127. = ;
3. w mantysie pomija się jedynkę, która stoi przed przecinkiem, dlatego mantysa wygląda następująco: 1000 0000 0000 0000 0000 000;
4. liczba jest ujemna, dlatego bit znaku przyjmuje wartość 1.

Ostatecznie: =

Podwójna precyzja: znak 1 bit, wykładnik 11 bitów, mantysa 52 bity.

Obliczenia wykonuje się w niemalże identyczny sposób. W obliczaniu wykładnika nie będzie się dodawać 127, lecz 1023. Stąd wykładnik wynosi: -1 + 1023 = 1022.

=

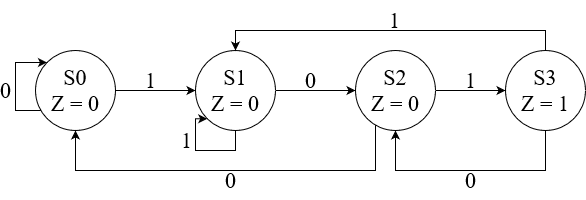
Ostatecznie:

=

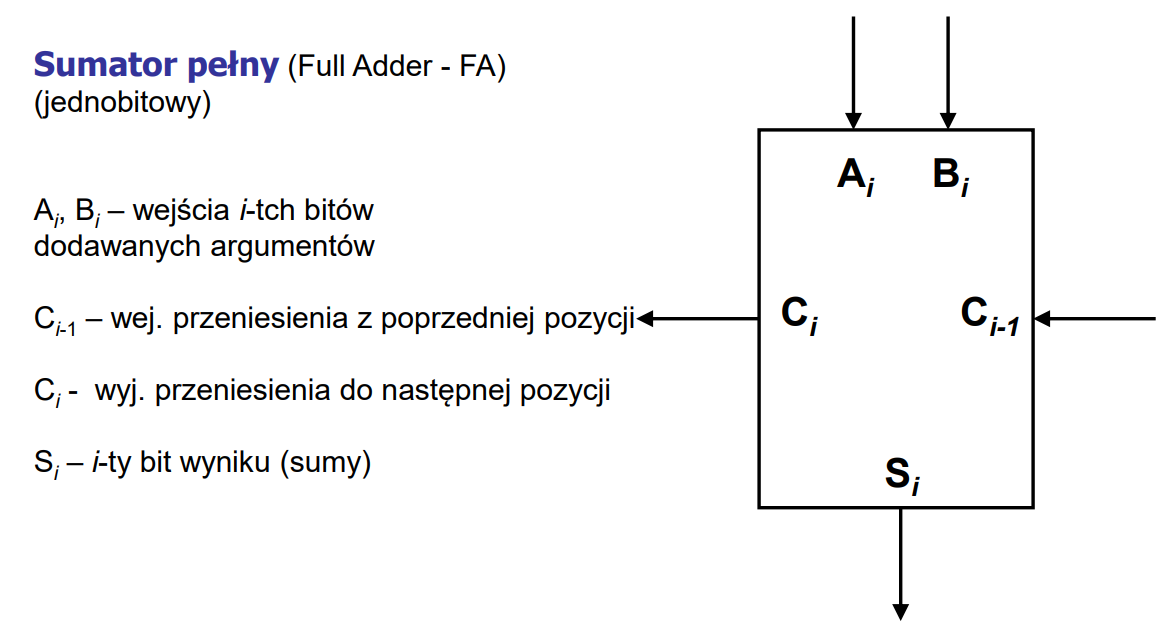
1. **Oblicz wartość liczby w systemie dziesiętnym zapisanej w kodzie uzupełnień do 2, .**

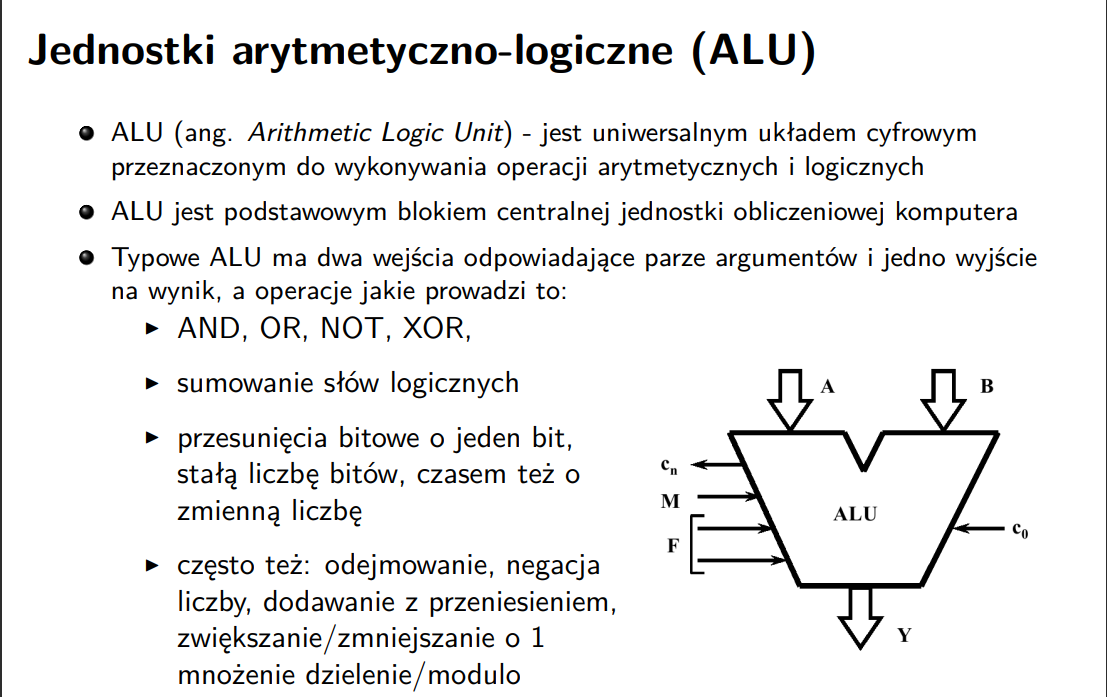
Obliczenia:

1. określenie znaku liczby. Pierwszą liczbą jest 1, czyli liczba jest ujemna;
2. negacja bitów. Otrzymana liczba: 00101;
3. dodanie 1. Otrzymana liczba: 00110;
4. przeliczenie na system dziesiętny. Otrzymana liczba: -6.
5. **Zaprojektować synchroniczny układ cyfrowy, który w ciągu zero-jedynkowym, przyłożonym na wejście X, rys. 1, ma wykryć sekwencję bitów (101), co sygnalizuje impulsem na wyjściu, Z=1. Po wykryciu sekwencji układ nie jest resetowany. Stany na wejściu X mogą się zmieniać jedynie między impulsami taktującymi (zegara).**



1. **Zaprojektuj blok logiczny zamieniający sumator na jednostkę arytmetyczno-logiczną.**

NIESKOŃCZONE



1. **Określ sieć bramek AND i OR realizującą funkcję f(a, b, c, d)=∑m(1, 5, 6, 10, 13, 14).**

Kroki potrzebne do rozwiązania zadania:

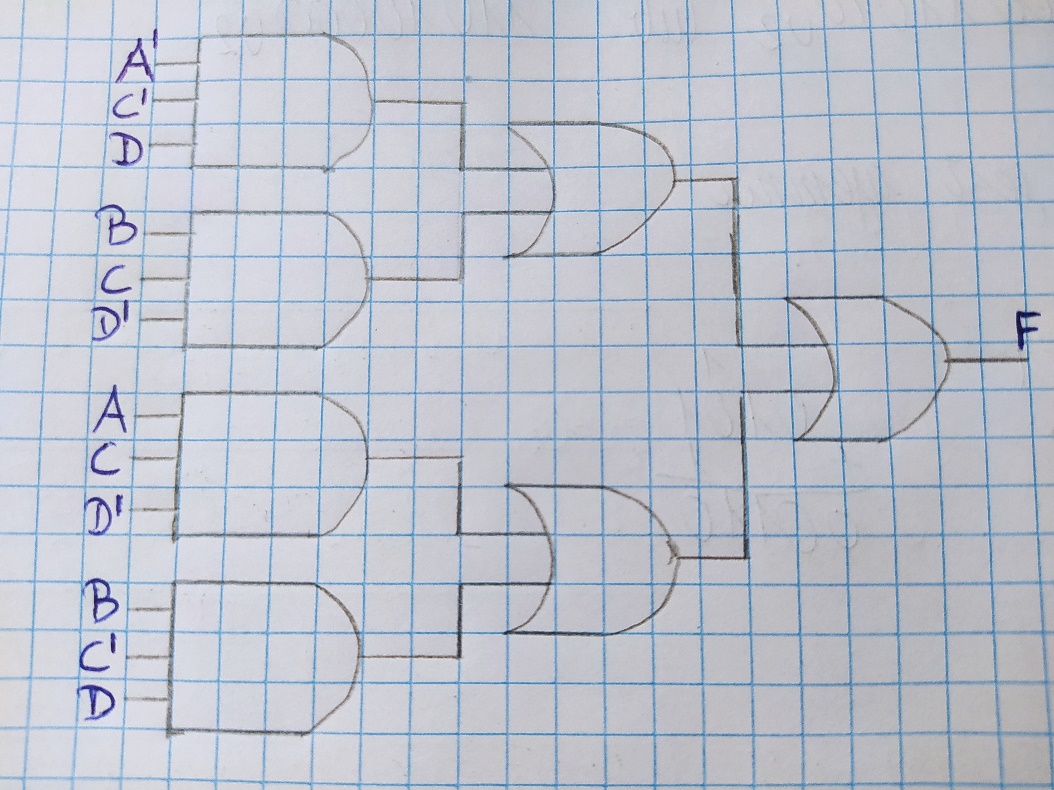
1. lista mintermów:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Minterm** | **a** | **b** | **c** | **d** | **Pozycja** |
|  | 0 | 0 | 0 | 1 | (00, 01) |
|  | 0 | 1 | 0 | 1 | (01, 01) |
|  | 0 | 1 | 1 | 0 | (01, 10) |
|  | 1 | 0 | 1 | 0 | (10, 10) |
|  | 1 | 1 | 0 | 1 | (11, 01) |
|  | 1 | 1 | 1 | 0 | (11, 10) |

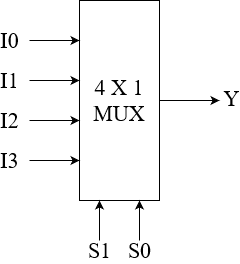
1. metoda Karnaugha

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ab/cd** | **00** | **01** | **11** | **10** |
| **00** | 0 | 1 | 0 | 0 |
| **01** | 0 | 1 | 0 | 1 |
| **11** | 0 | 1 | 0 | 1 |
| **10** | 0 | 0 | 0 | 1 |

f(a, b, c, d) = a'c'd + bcd' + acd' + bc'd

1. sieć bramek AND i OR
2. **Zaprojektuj multiplekser 4-wejściowy.**

Rysunek multipleksera:

 n = 4 – liczba wejść

m = – liczba sygnałów sterujących

m = = 2

Tabela:

|  |  |  |
| --- | --- | --- |
|  |  | **Y** |
| 0 | 0 |  |
| 0 | 1 |  |
| 1 | 0 |  |
| 1 | 1 |  |

Równanie:

Y = + + +

1. **Zrealizuj następujące funkcje wyjściowe bazując na ROM (8x4). Y0(A, B, C)=∑m(0, 2, 5);  
   Y1(A, B, C)=∑m(1, 3, 4); Y2(A, B, C)=∑m(2, 4, 7); Y3(A, B, C)=∑m(0, 1, 3, 4, 5).**

Przeliczenie cyfr z systemu dziesiętnego na dwójkowy:

=

=

=

=

=

=

=

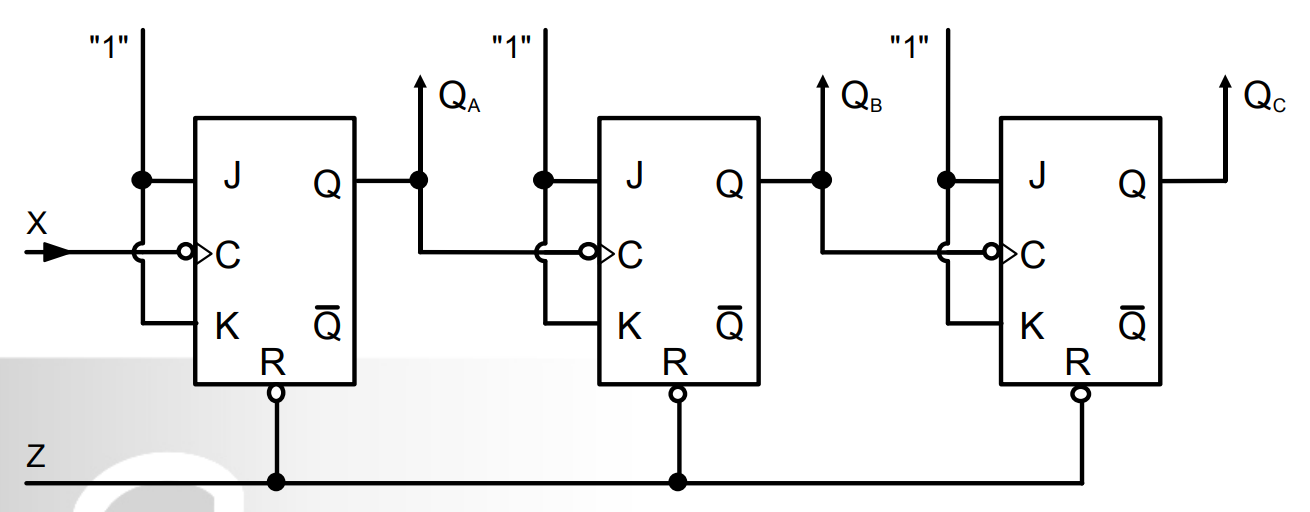
Tablica prawdy ROM (8x4) dla zdefiniowanych funkcji wyjściowych:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Nr adresu** | **A** | **B** | **C** |  |  |  |  |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 4 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 5 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 6 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

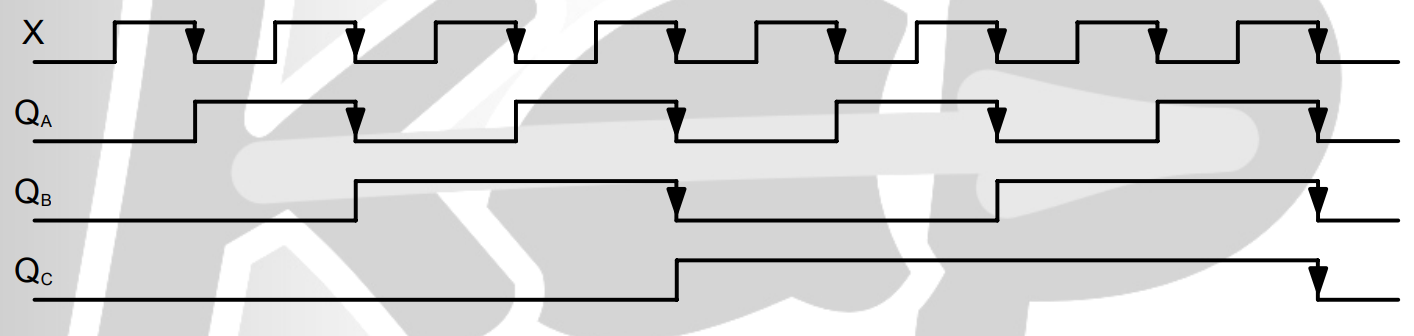
Y przyjmuje wartość 1, jeśli zawiera w sobie liczbę odpowiadającą adresowi.

1. **Zaprojektuj licznik szeregowy zliczający w przód do 8 na przerzutnikach JK.**

Licznikiem nazywamy sekwencyjny układ cyfrowy służący do zliczania i pamiętania liczby impulsów podawanych w określonym przedziale czasu na jego wejście zliczające.

Schemat:

Przebiegi czasowe:



Zliczane impulsy są wprowadzone na wejście zegarowe (x) pierwszego przerzutnika. Wejścia zegarowe kolejnych przerzutników są zwarte z wyjściami Q poprzednich przerzutników. Podanie zera logicznego na wejścia zerujące (R) przerzutników (z=0) umożliwia asynchroniczne wyzerowanie licznika w dowolnej chwili, w czasie jego pracy.

1. **Przeanalizować, czy w układzie kombinacyjnym opisanym funkcją przełączającą   
   y = x1x2 ′ + x2x3 wystąpi hazard statyczny.**

Tabela:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | **y** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Mapa Karnaugha:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **/** | **00** | **01** | **11** | **10** |
| **0** | 0 | 0 | 1 | 0 |
| **1** | 1 | 1 | 1 | 0 |

Analiza możliwego hazardu:

Hazard statyczny na mapie Karnaugha występuje tam, gdzie sąsiednie komórki różniące się jednym bitem nie są objęte tą samą grupą. Eliminuje się go poprzez dodanie dodatkowej grupy. Jeśli obok siebie istnieje grupa A oraz grupa B i można ich część połączyć dodatkową grupą to lepiej to zrobić. Nie można zapominać o tym, że siatki Karnaugha się zwijają.

W tym przypadku zmiana wektora wejść z 111 na 101 może spowodować przejściowo poziom niski na wyjściu. Hazard statyczny występuje w tym układzie.

Wzór funkcji, który uwzględnia eliminację hazardu: **y =** **x1x2 ′ + x2x3 + x1x3**

1. **Zaprojektuj detektor błędu dla kodu binarnego 6-3-1-1 liczb dziesiętnych. Wyjście F jest 1, gdy na wejściach A, B, C, D jest błąd.**

Kod 6-3-1-1 wykorzystuje 4 bity (A, B, C, D), w których wartości mają następujące wagi:

- bit A ma wagę 6,

- bit B ma wagę 3,

- bit C ma wagę 1,

- bit D ma wagę 1.

Kod ten ma odpowiadać liczbie dziesiętnej w zakresie 0 – 9.

Przykład: 1001 = 6 \* 1 + 3 \* 0 + 1 \* 0 + 1 \* 1 = 6 + 1 = 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **Liczba dziesiętna** | **Funkcje** |
| 0 | 0 | 0 | 0 | 0 | ​ = \* \* \* |
| 0 | 0 | 0 | 1 | 1 | = \* \* \* D |
| 0 | 0 | 1 | 1 | 2 | = \* \* C \* D |
| 0 | 1 | 0 | 0 | 3 | = \* B \* \* |
| 0 | 1 | 0 | 1 | 4 | = \* B \* \* D |
| 0 | 1 | 1 | 1 | 5 | = \* B \* C \* D |
| 1 | 0 | 0 | 0 | 6 | = A \* \* \* |
| 1 | 0 | 0 | 1 | 7 | = A \* \* \* D |
| 1 | 0 | 1 | 1 | 8 | = A\* \* C \* D |
| 1 | 1 | 0 | 0 | 9 | = A \* B\* \* |

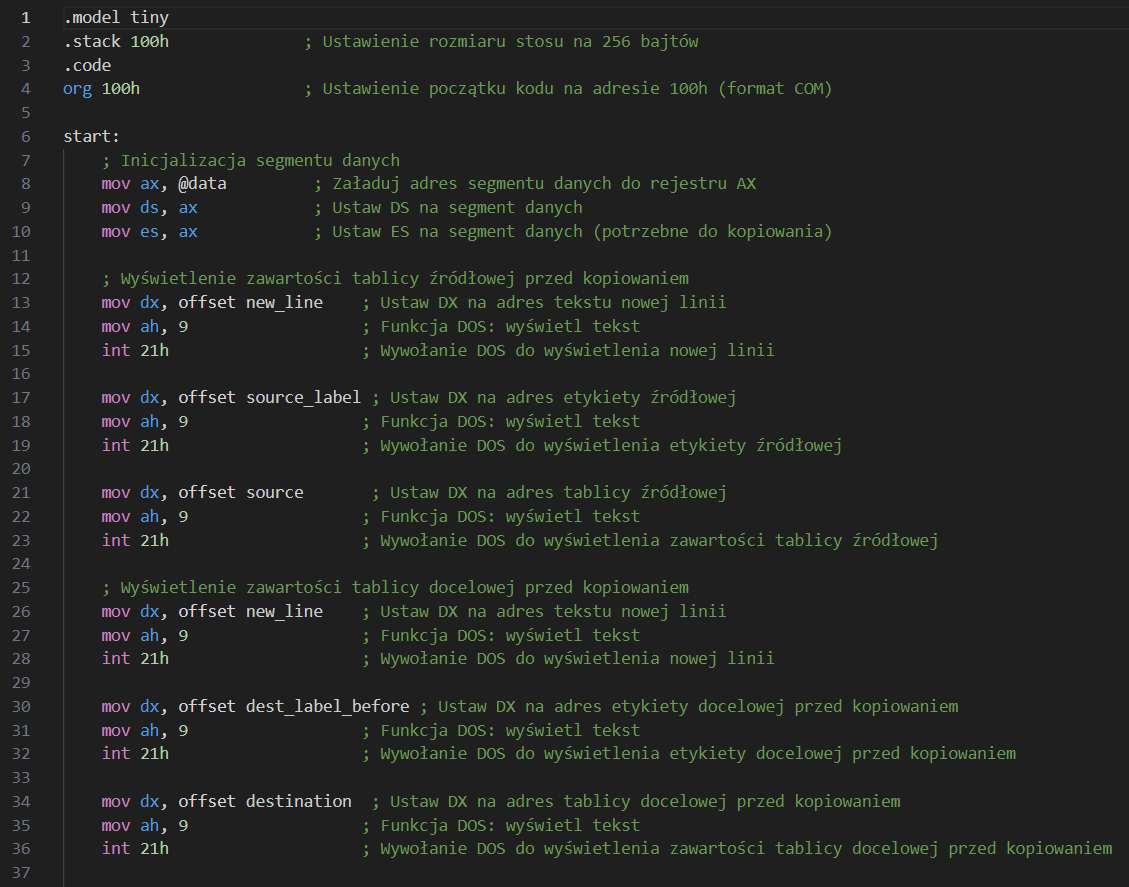
Detektor błędu:

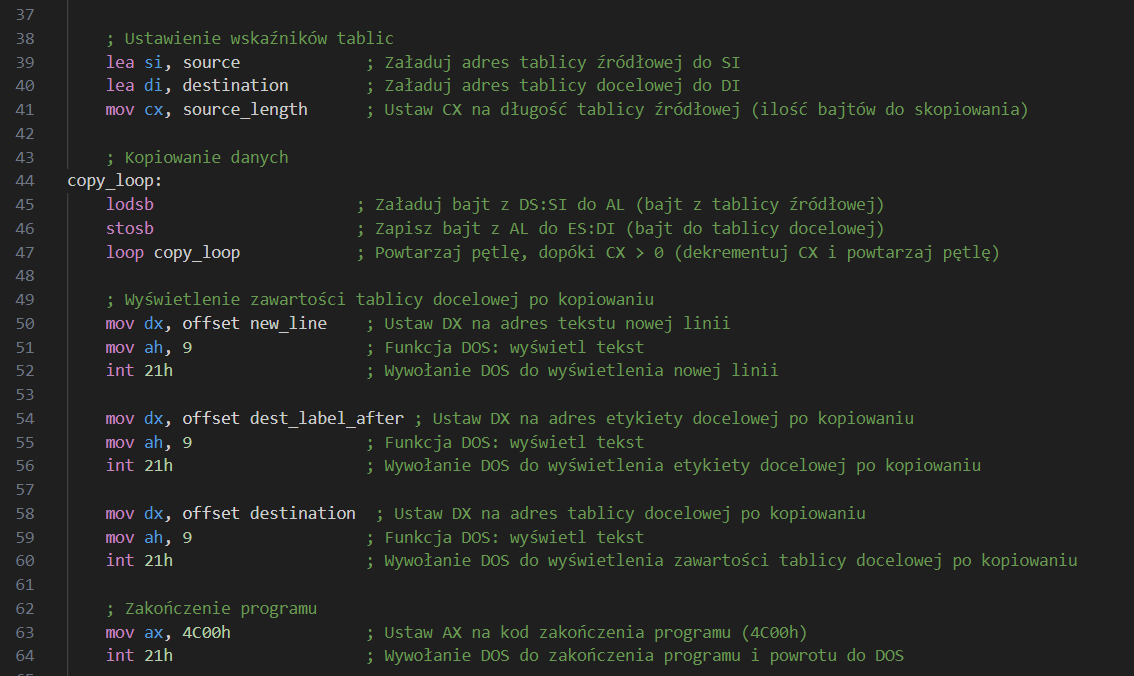
F =

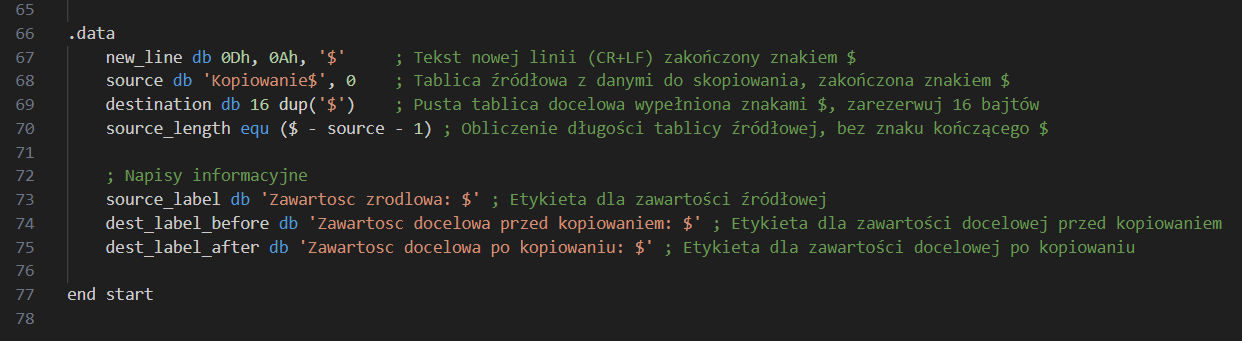
**Architektura systemów komputerowych**

**3. Napisz program realizujący kopiowanie danych zdefiniowanych w postaci tablic i realizowanych   
w pętli. Omów wykorzystanie rejestrów procesora. Odnieś się do sposobu realizacji programu   
w zależności od jego rozmiaru (modelu). Jaka jest optymalność tego programu. Jak rozumiesz pojęcie adresu i wskaźnika.**

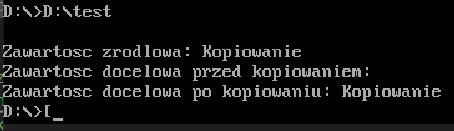
Kod programu:







Działanie programu:



Wykorzystanie rejestrów:

**- AX** do operacji systemowych i przerwań,

**- DS/ES** do zarządzania segmentami danych,

**- SI/DI** do indeksowania tablic,

**- CX** jako licznik pętli,

**- DX** do przechowywania adresów tekstów,

**- AL** do manipulacji pojedynczymi bajtami.

Sposób realizacji programu w zależności od jego rozmiaru (modelu):

W modelu tiny wszystko odbywa się w jednym segmencie, co maksymalnie upraszcza obsługę pamięci, ale jednocześnie ogranicza rozmiar programu.

Optymalność programu:

Program jest **optymalny pod względem użycia pamięci**, ponieważ zajmuje mało miejsca i nie wymaga przełączania segmentów. Pętla z użyciem rejestru **CX** (LOOP copy\_loop) jest wydajna, ponieważ operacja LOOP automatycznie zmniejsza wartość CX i sprawdza, czy należy kontynuować pętlę. Dzięki temu unika się dodatkowych operacji porównania.

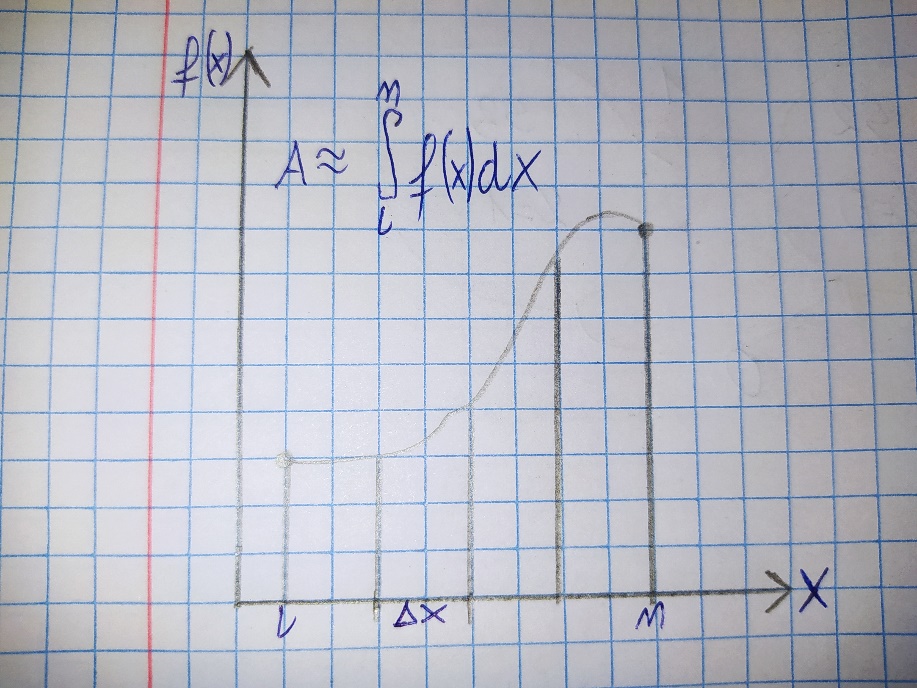
Pojęcie adresu i wskaźnika:

**Adres** to miejsce w pamięci, które jest identyfikowane przez unikalny numer. W kodzie asemblera adresy są używane do lokalizowania danych i instrukcji w pamięci.

**Wskaźnik** to zmienna lub rejestr, który przechowuje adres pamięci. Wskaźnik wskazuje na miejsce w pamięci, a operacje mogą być wykonywane na danych znajdujących się pod tym adresem.

**Metody numeryczne**

**10. Opisz i narysuj stosowne ilustracje z oznaczeniami dla metody trapezów i Simpsona przy podziale przedziału całkowania na podprzedziały.**

a) metoda trapezów

Ogólny wzór na pole trapezu:

P = \* (a+b) \* h

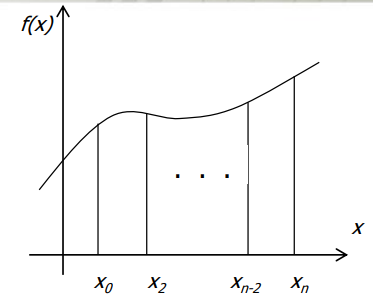
Dla wzoru metody trapezów:

h = x

a = f()

b = f()

P = x[(f() + f()) + 2]

b) metoda Simpsona

= [f() + 4 + 2 + f()]

i = np – „i” ma być nieparzyste

i = p – „i” ma być parzyste