Języki asemblerowe

WYKŁAD 4

Dr Krzysztof Balicki

Wybrane funkcje przerwania 10h

- AH=00h – ustawienie trybu pracy karty graficznej
- AH=01h – zdefiniowanie rozmiaru kursora
- AH=02h – ustawienie pozycji kursora na ekranie
- AH=03h – odczytanie położenia kursora na ekranie
- AH=05h – zmiana strony aktywnej
- AH=06h – przewijanie strony aktywnej w górę

Wybrane funkcje przerwania 10h

- AH=07h – przewijanie strony aktywnej w dół
- AH=08h – odczytanie znaku i atrybutu w miejscu ustawienia kursora
- AH=09h – zapisanie znaku i atrybutu w miejscu ustawienia kursora
- AH=0Ah – zapisanie znaku bez atrybutu w miejscu ustawienia kursora
- AH=0Bh – wybór palety kolorów
- AH=0Ch – wyświetlenie punktu

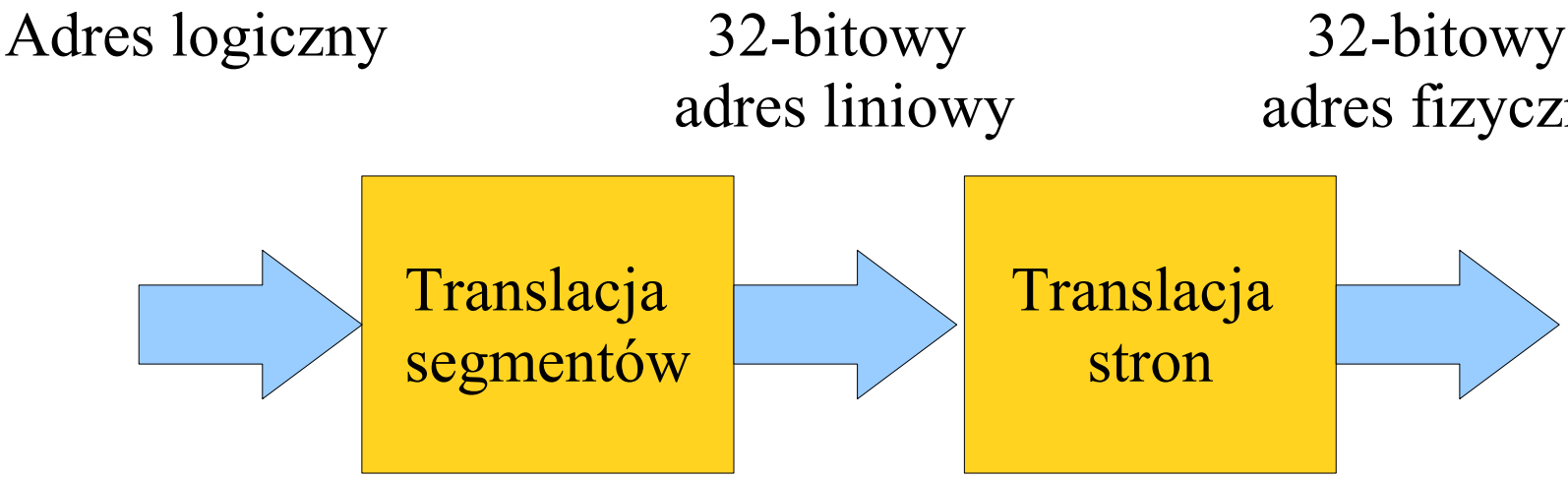
Wybrane funkcje przerwania 10h

- AH=0Dh – odczytanie punktu
- AH=0Fh – odczytanie stanu ekranu
- AH=09h – zapisanie znaku i atrybutu w miejscu ustawienia kursora
- AH=0Ah – zapisanie znaku bez atrybutu w miejscu ustawienia kursora
- AH=0Bh – wybór palety kolorów
- AH=0Ch – wyświetlenie punktu

Tryby adresowania pamięci

- Tryb rzeczywisty
  - adres 16 bitowy
  - możliwość uruchamiania programów napisanych dla 8086
- Tryb chroniony
  - adres 32 bitowy
  - wspierane są segmentacja i stronicowanie

Tryb chroniony



Przy braku translacji stron adres liniowy jest adresem fizycznym

Tryb chroniony

- Przy translacji adresu logicznego na adres liniowy offset jest dodawany do 32 bitowego adresu bazowego.
- Offset może być 16 lub 32 bitowy.

Translacja segmentów

- Każdy rejestr segmentowy posiada część:
  - „widoczną” - 16 starszych bitów - selektor segmentu,
  - „niewidoczną” - 16 młodszych bitów - część ładowana automatycznie przez procesor z tablicy deskryptorów.

Translacja segmentów

- Selektor segmentu zawiera:
  - indeks (13 bitów):
    - wybiera deskryptor segmentu z lokalnej (dla danego programu) lub globalnej (dostępnej w całym systemie) tablicy deskryptorów,
    - umożliwia wybór spośród 8192 deskryptorów,
    - każdy deskryptor jest 8 bajtowy, procesor mnoży indeks przez 8 i dodaje do adresu bazowego wybranej tablicy deskryptorów,
  - wskaźnik tablicy (1 bit):
    - określa, która tablica jest wybrana: lokalna czy globalna,
  - poziom priorytetu dostępu (2 bity):
    - większa wartość → mniejszy priorytet

Translacja segmentów

- Deskryptor segmentu opisuje własności segmentu:
  - 32 bitowy adres bazowy segmentu w 4GB fizycznej przestrzeni adresowej,
  - 20 bitowy rozmiar segmentu,

Translacja segmentów

- Deskryptor segmentu opisuje własności segmentu (cd.):
  - informacje kontrolne i statusowe, np.:
    - granularność - określa czy rozmiar segmentu jest definiowany w bajtach czy w jednostkach 4KB,
    - bit D/B:
      - w sekcji kodu: rozmiar operandów i offsetu (16 lub 32 bity),
      - w sekcji danych: rozmiar stosu (FFFFH lub FFFFFFFFH),
    - bit S - określa czy segment jest systemowy czy aplikacyjny
    - opis poziomu priorytetu dostępu
    - typ segmentu: kod czy dane

Tryb rzeczywisty

- Pamięć jest zbiorem segmentów.
- Każdy segment posiada rozmiar 64 KB.
- Miejsce w pamięci specyfikowane jest przez dwie składowe stanowiące adres logiczny:
  - adres bazowy segmentu (określający adres początkowy segmentu w pamięci),
  - offset - adres efektywny (określający adres względny w ramach danego segmentu).
- Adres bazowy jest 20 bitowy - w rejestrze przechowywane jest 16 starszych bitów, dla 4 młodszych bitów przyjmuje się wartości 0.

Tryb rzeczywisty

- Segmenty mogą zaczynać się w pamięci tylko w miejscach, których adresy mają zerowe cztery najmłodsze bity.

Tryby adresowania operandów

- Operandy rozkazów mogą być umieszczone w:
  - rejestrach,
  - w instrukcjach,
  - w pamięci głównej (zazwyczaj w segmencie danych),
  - w portach wejścia/wyjścia.

Tryby adresowania operandów

- Rejestrowy tryb adresowania
  - dane umieszczone są w rejestrach, np.:

    mov AX, BX
  - jest najbardziej efektywnym trybem adresowania
- Natychmiastowy tryb adresowania
  - dane umieszczone są w segmencie kodu, a nie w segmencie danych, np.:

    mov AX, 65h

Tryby adresowania operandów

- Bezpośredni tryb adresowania
  - dane umieszczone są w segmencie danych
  - do lokalizacji danych potrzebny jest bazowy adres segmentu i offset w ramach segmentu
  - bazowy adres segmentu jest najczęściej umieszczany w rejestrze DS
  - offset jest specyfikowany bezpośrednio przez instrukcję
  - w programie asemblerowym offset wskazywany jest przez nazwę zmiennej opisującej dane
  - ten tryb można zastosować tylko do jednego operandu

Tryby adresowania operandów

- Pośredni tryb adresowania
  - dane umieszczone są w segmencie danych
  - offset umieszczony jest w jednym z rejestrów
  - na rejestrze zawierającym offset mogą być wykonywane różne operacje

Reprezentacja liczb

- Liczby wprowadzane z klawiatury traktowane są jako ciągi znaków.
- Liczby mogą zostać zamienione na postać binarną lub przetwarzane w postaci dziesiętnej.
- W przypadku postaci dziesiętnej liczby mogą być reprezentowane w kodzie ASCII lub w kodzie BCD.

Konwersja ciągu cyfr (liczby) na postać binarną

```
znak = czytaj_znak_liczby
liczba = znak − ’0’
i = ilosc_cyfr -1
do
      znak = czytaj_znak_liczby
      liczba =liczba * 10 + (znak − ’0’)
      i = i − 1
while (i > 0)
```

Reprezentacja liczb w kodzie ASCII

- Liczba jest przechowywana jako ciąg kodów ASCII jej cyfr.
- Kody ASCII cyfr:
  - 0 - 30h
  - 1 - 31h
  - 2 - 32h
  - 3 - 33h
  - 4 - 34h
  - 5 - 35h
  - 6 - 36h
  - 7 - 37h
  - 8 - 38h
  - 9 - 39h

Reprezentacja liczb w kodzie BCD

- Reprezentacja upakowana
  - każda cyfra liczby przechowywana jest na 4 bitach
  - jeden bajt zawiera dwójkowe reprezentacje dwóch cyfr dziesiętnych liczby
  - przykład:
    - liczba 2356 przechowywana jest w dwóch bajtach:

      23h 56h

Reprezentacja liczb w kodzie BCD

- Reprezentacja nieupakowana
  - każda cyfra liczby przechowywana jest na 8 bitach
  - jeden bajt zawiera dwójkową reprezentację jednej cyfry dziesiętnej liczby
  - przykład:
    - liczba 2356 przechowywana jest w czterech bajtach:

      02h 03h 05h 06h

Operacje na liczbach reprezentowanych w kodzie ASCII

- Przy operacjach arytmetycznych na liczbach reprezentowanych w kodzie ASCII używane są rozkazy poprawek:
  - aaa - poprawka przy dodawaniu
  - aas - poprawka przy odejmowaniu
  - aam - poprawka przy mnożeniu
  - aad - poprawka przy dzieleniu

Operacje na liczbach reprezentowanych w kodzie BCD

- Przy operacjach arytmetycznych na liczbach reprezentowanych w upakowanym kodzie BCD używane są rozkazy poprawek:
  - daa - poprawka przy dodawaniu
  - das - poprawka przy odejmowaniu
- Brak jest rozkazów wspierających mnożenie i dzielenie