Ćwiczenie nr 3

Wyświetlanie i wczytywanie danych

3.1 Wstęp

Współczesne komputery przetwarzają dane zakodowane za pomocą ciągów zerojedynkowych. W szczególności przetwarzane liczby kodowane są w systemie dwójkowym. Ponieważ człowiek posługuje się systemem dziesiętnym zachodzi więc konieczność każdorazowego przetworzenia danych liczbowych na postać binarną przy wprowadzaniu ich do komputera i odwrotnie przy wyprowadzaniu danych. Okazuje się, że algorytmy używane do takiej zamiany są stosunkowo proste, i dają się łatwo zakodować na poziomie instrukcji procesora. Niniejsza instrukcja omawia podstawowe algorytmy używane przy wprowadzaniu i wyprowadzaniu danych liczbowych.

3.2 Zamiana liczby binarnej na dziesiętną

Na poziomie programu, uzyskanie reprezentacji dziesiętnej pewnej liczby binarnej sprowadza się w istocie do uzyskania cyfr dziesiętnych tej liczby, zakodowanych w postaci znaków ASCII. W tym celu daną liczbę dzieli się wielokrotnie przez 10, a uzyskane reszty, po konwersji na znaki ASCII, stanowią poszukiwane cyfry. Wyjaśnimy to na przykładzie.

Dzielna	Reszta
27409	9
2740	0
274	4
27	7
2	2
0	

Wprawdzie podany przykład dotyczy liczb dziesiętnych, ale identyczne wartości uzyskuje się dla liczb zapisanych w systemie dwójkowym – wartość ilorazu i reszty z dzielenia nie zależy bowiem od podstawy systemu w jakim zostały zapisane liczby! Zauważmy, że po pierwszym dzieleniu uzyskaliśmy cyfrę jednostek, w drugim dzieleniu cyfrę dziesiątek, w trzecim dzieleniu cyfrę setek, itd.

Uzyskane reszty można łatwo zamienić na cyfry w kodzie ASCII. Kody przyporządkowane kolejnym cyfrom podane są w poniższej tabelce.

Cyfra	Kod ASCII
0	30H
1	31H
2	32H
3	33H
4	34H
5	35H
6	36H
7	37H
8	38H
9	39H

Nietrudno zauważyć, że kod ASCII cyfry jest o 30H większy od wartości cyfry. Zatem zamiana na kod ASCII polega po prostu na dodaniu liczby 30H.
Na poziomie instrukcji procesora dzielenie wykonywane jest za pomocą instrukcji:

- `div` dla liczb bez znaku,
- `idiv` dla liczb ze znakiem.

Obie te instrukcje mają kilka odmian różniących się rozmiarem dzielnej i dzielnika. Przed wykonaniem dzielenia dzielna musi się znajdować w ustalonych (niżej podanych) rejestrach. Dzielnik może się znajdować w jednym z rejestrów ogólnego przeznaczenia albo w postaci 1-, 2- lub 4-bajtowej liczby umieszczonej w komórkach pamięci. Iloraz i reszta z dzielenia umieszczane są w ustalonych rejestrach. Dzielna, dzielnik, iloraz i reszta spełniają poniższą zależność

$$\text{dzielna} = \text{iloraz} \cdot \text{dzielnik} + \text{reszta}$$

przy czym znak reszty jest taki sam jak znak dzielnej. Jednocześnie wartość bezwzględna reszty jest zawsze mniejsza od wartości bezwzględnej dzielnika. Zestawienie operandów i wyników instrukcji dzielenia `div` i `idiv` zawiera poniższa tabela.

Rodzaj	**Dzielna**	**Dzielnik**	**Iloraz**	**Reszta**
64b./32b.	EDX (starsza część) i EAX (młodsza część)	rejestr 32-bitowy albo liczba 4-bajtowa w pamięci	EAX	EDX
32b./16b.	DX (starsza część) i AX (młodsza część)	rejestr 16-bitowy albo liczba 2-bajtowa w pamięci	AX	DX
16b./8b.	AX	rejestr 8-bitowy albo liczba 1- bajtowa w pamięci	AL	AH

Przykładowo, instrukcja

```
div bx
```

wykonuje dzielenie liczby 32-bitowej bez znaku umieszczonej w rejestrach DX i AX przez liczbę 16-bitową bez znaku umieszczoną w rejestrze BX. Iloraz z dzielenia zostaje wpisany do rejestru AX, a reszta z dzielenia do rejestru DX.

Przypuśćmy, że zamierzamy wyświetlić na ekranie w postaci dziesiętnej liczbę 16-bitową, która znajduje się w rejestrze AX. Jak to wcześniej opisano, konwersja wymaga wielokrotnego dzielenia podanej liczby przez 10. Czy instrukcja „16b./8b." będzie tu odpowiednia? Instrukcja ta dzieli liczbę 16-bitową przez 8-bitową, więc wydaje się doskonale pasować do postawionego zadania. Rozpatrzmy jednak prosty przykład. Jeśli w rejestrze AX znajduje się liczba 4705, a w rejestrze BL liczba 10, to w wyniku zastosowania instrukcji dzielenia

```
div bl
```

uzyskamy iloraz: 4705/10 = 470 i resztę: 4705 mod 10 = 5. Zauważmy jednak, że dla podanej instrukcji iloraz wpisywany jest do rejestru AL. Ponieważ AL jest rejestrem 8-bitowym, więc mogą być w nim przechowywane liczby bez znaku z zakresu 0–255. Zatem iloraz nie da się zapisać w rejestrze AL – pojawi się nadmiar! Konsekwencją wystąpienia nadmiaru przy dzieleniu jest wyjątek generowany przez procesor, co praktycznie oznacza zakończenie wykonywania programu.

Tak więc wybrany rodzaj instrukcji dzielenia jest nieodpowiedni. Trzeba zatem wybrać dzielenie „32b./16b.”, w którym iloraz zapisywany jest w postaci liczby 16-bitowej i może przyjmować wartości z przedziału 0–65535. W tym jednak przypadku dzielna jest 32-bitowa. Jednak zamiana 16-bitowej liczby bez znaku na liczbę 32-bitową sprowadza się po prostu do wyzerowania najstarszych 16 bitów.

Na podstawie podanych wyjaśnień można już napisać fragment programu wyznaczający kolejne cyfry dziesiętne. Ponieważ maksymalna wartość liczby w rejestrze AX wynosi 65535, więc w trakcie konwersji należy uzyskać 5 cyfr dziesiętnych. Omawiane czynności realizuje poniższy fragment programu.

```
;wyświetlanie liczby całkowitej zawartej w AX
;uwaga: poniższy fragment wymaga korekty (zob. dalszy opis)!

      mov   cx, 5       ;liczba obiegów pętli
      mov   bx, 10      ;dzielnik
p1:   mov   dx, 0       ;zerowanie starszej części dzielnej
      div   bx          ;dzielenie przez 10 - iloraz w AX, reszta w DX
      add   dx, 30      ;zamiana reszty na kod ASCII
      push  ax          ;przechowanie ilorazu na stosie
      mov   ah, 2
      int   21H         ;wyświetlenie cyfry na ekranie
      pop   ax          ;przywrócenie ilorazu w AX
      loop  p1          ;sterowanie pętlą
```

Niestety, podany fragment nie będzie wyświetlał poprawnych wartości. Przykładowo, jeśli w rejestrze AX znajdować się będzie liczba 0100000000000000 (dziesiętnie 16384), to na ekranie pojawi się liczba 48361. Przyczyny uzyskania takiego wyniku są oczywiste: zastosowany algorytm generuje cyfry począwszy od jednostek, potem dziesiątek, setek, itd. I w takiej też kolejności cyfry wyświetlane są na ekranie.

Istnieje kilka różnych sposobów poprawienia podanego programu. Jeden z tych sposobów wymaga magazynowania uzyskiwanych cyfr w tablicy. Po uzyskaniu wszystkich pięciu cyfr są one pobierane z tablicy w odpowiedniej kolejności i wyświetlane. Skorygowana wersja programu podana jest poniżej.

```
; wyświetlanie liczby całkowitej zawartej w AX
dane  SEGMENT
      cyfry db 5 dup (?)  ;zdefiniowanie tablicy 5-cio bajtowej
…
      mov   bx, SEG dane
      mov   ds, bx
      mov   cx, 5         ;liczba obiegów pętli
      mov   bx, 10        ;dzielnik
      mov   si, 4         ;indeks początkowy w tablicy cyfry
p1:   mov   dx, 0         ;zerowanie starszej części dzielnej
      div   bx            ;dzielenie przez 10 - iloraz w AX, reszta w DX
      add   dx, 30H       ;zamiana reszty na kod ASCII
      mov   cyfry[si], dl ;odesłanie kodu ASCII kolejnej cyfry do tablicy
      dec   si
      loop  p1

      mov   cx, 5         ;wyświetlanie cyfr, liczba obiegów pętli
      mov   si, 0         ;indeks początkowy w tablicy cyfry
p2:   mov   dl, cyfry[si] ;pobranie kodu ASCII kolejnej cyfry
      mov   ah, 2         ;wyświetlenie cyfry na ekranie
      int   21H
      inc   si            ;zwiększenie indeksu w rejestrze SI o 1
      loop  p2            ;sterowanie pętlą wyświetlania
```

Przedstawione tu rozwiązania ma ciągle pewne wady. Wprawdzie liczba 0100000000000000 będzie wyświetlana poprawnie jako 16384, to jednak liczba 0000000011001111 (dziesiętnie 207) będzie wyświetlana w postaci 00207. W wyświetlanej liczbie zamiast dwóch pierwszych zer lewej strony powinny być umieszczone spacje. Zamiana na spacje nie może jednak obejmować wszystkich zer w liczbie, ale tylko zera z lewej strony przed pierwszą cyfrą niezerową. Podany niżej fragment programu zawiera także operacje usuwania zer nieznaczących z lewej strony.

```
; wyświetlanie liczby całkowitej zawartej w AX
dane  SEGMENT
      cyfry db 5 dup (?)
…
      mov   bx, SEG dane
      mov   ds, bx
      mov   cx, 5          ;liczba obiegów pętli
      mov   bx, 10         ; dzielnik
      mov   si, 4          ;indeks początkowy w tablicy tbl_cyfr
p1:   mov   dx, 0          ;zerowanie starszej części dzielnej
      div   bx             ;dzielenie przez 10; iloraz w AX, reszta w DX
      add   dx, 30H        ;zamiana reszty na kod ASCII
      mov   cyfry[si], dl ;odesłanie kodu ASCII kolejnej cyfry do tablicy
      dec   si             ;zmniejszenie indeksu w rejestrze SI o 1
      loop  p1             ;sterowanie pętlą

      ;usuwanie zer nieznaczących z lewej strony
      mov   cx, 4          ;liczba obiegów pętli usuwania zer nieznaczących
      mov   si, 0          ;indeks początkowy w tablicy cyfry
p2:   cmp   byte PTR cyfry[si], 30H ;czy cyfra '0'
      jne   druk
      mov   byte PTR cyfry[si], 20H ;wpisanie kodu spacji w miejsce zera
      inc   si
      loop p2

      ;wyświetlanie cyfr
druk: mov   cx, 5          ;liczba obiegów pętli
      mov   si, 0          ;indeks początkowy w tablicy cyfry
p3:   mov   dl, cyfry[si] ;pobranie kodu ASCII kolejnej cyfry
      mov   ah, 2          ;wyświetlenie cyfry na ekranie
      int   21H
      inc   si             ;zwiększenie indeksu w rejestrze SI o 1
      loop  p3             ;sterowanie pętlą wyświetlania
```

Rozpatrzymy teraz nieco inne rozwiązanie, w którym uzyskiwane cyfry będą gromadzone na stosie. Zaletą poniższego rozwiązania jest także to, że nie powstają tu zera nieznaczące z lewej strony.

```
      mov   cx, 0       ;licznik cyfr
      mov   bx, 10      ;dzielnik
p1:   mov   dx, 0       ;zerowanie starszej części dzielnej
      div   bx          ;dzielenie przez 10 – iloraz w AX, reszta w DX
      add   dx, 30H     ;zamiana reszty na kod ASCII
      push  dx          ;zapisanie cyfry na stosie
      inc   cx          ;inkrementacja licznika cyfr
      cmp   ax, 0       ;porównanie uzyskanego ilorazu
      jnz   p1          ;skok gdy iloraz jest różny od zera

p2:   pop   dx          ;pobranie kodu ASCII kolejnej cyfry
      mov   ah, 2
      int   21H         ;wyświetlenie cyfry na ekranie
      loop  p2          ; sterowanie pętlą wyświetlania
```

3.3 Zamiana liczby dziesiętnej na binarną

W poprzednio omówionym algorytmie konwersji na postać binarną stosowane było wielokrotne dzielenie przez 10. Z kolei przy zamianie z postaci dziesiętnej na binarnej wykonuje się wielokrotnie mnożenie przez 10. Wyjaśnimy to dokładniej na przykładzie.

Przypuśćmy, że użytkownik programu napisał na klawiaturze liczbę 30982. Z punktu widzenia programu oznacza to, że zostały wprowadzone kody ASCII cyfr 3, 0, 9, 8, 2. W celu uzyskania postaci binarnej liczby trzeba wykonać obliczenie (na liczbach binarnych za pomocą rozkazów komputera):

$$3 \cdot 10000 + 0 \cdot 1000 + 9 \cdot 100 + 8 \cdot 10 + 2 = (((((0 \cdot 10 + 3) \cdot 10 + 0) \cdot 10 + 9) \cdot 10 + 8) \cdot 10 + 2)$$

Prawa strona powyższego wyrażenia wskazuje na możliwość zastosowania pętli: w każdym obiegu pętli należy mnożyć dotychczas uzyskany wynik przez 10 i dodać wartość kolejnej cyfry. Przyjmujemy, że początkowa wartość wyniku konwersji jest równa 0.
Poniższy fragment wczytuje liczbę dziesiętną bez znaku z klawiatury i wpisuje ją w postaci binarnej do rejestru AX. Zakładamy, że liczba nie przekracza 65535. Po wprowadzeniu wszystkich cyfr liczby użytkownik naciska klawisz `Enter`. Ponadto zakładamy, że użytkownik programu wprowadza wyłącznie znaki cyfr i nie używa klawisza `Backspace`.

```
; wczytywanie liczby do AX, zakończenie wczytywania po Enter
      mov   si, 0             ;początkowa wartość wyniku konwersji w SI
p1:   mov   ah, 1             ;wczytanie znaku w kodzie ASCII
      int   21H               ;z klawiatury do AL
      cmp   al, 13
      je    nacis_enter       ;skok gdy naciśnięto klawisz Enter
      sub   al, 30H           ;zamaiana kodu ASCII na wartość cyfry
      mov   bl, al            ;przechowanie kolejnej cyfry w AL
      mov   bh, 0             ;zerowanie rejestru BH
      mov   ax, 10            ;mnożnik
      mul   si                ;mnożenie dotychczas uzyskanego wyniku przez
                              ;10 iloczyn zostaje wpisany do rejestrów DX:AX
      add   ax, bx            ;dodanie aktualnie wczytanej cyfry
      mov   si, ax            ;przesłanie wyniku obliczenia do rejestru SI
      jmp   p1                ;
nacis_enter:
      mov   ax, si            ;przepisanie wyniku konwersji do rejestru AX
```