Katedra Inżynierii Komputerowej
Politechnika Częstochowska

Laboratorium Programowania niskopoziomowego

Laboratorium 7

7 listopada 2012

1 Suma liczb BCD

Należy utworzyć metodę, mnożącą dwie liczby BCD o określonej liczbie znaków. Liczba znaków ma być pobrana od użytkownika.

Wskazówki:

- przykład z wykładów,
- uwaga na kolejność sumowanych liczb BCD (sumujemy od najmniej znaczących do najbardziej znaczących - od końca tablicy),
- reprezentacja liczb BCD - poszczególne cyfry to liczby zapisane w tablicy znaków typu char,
- uwaga na przepełnienie - jeżeli sumujemy dwie liczby 2 cyfrowe, to musimy mieć tablice co najmniej 3 elementowe, do elementu zerowego tablicy przypisujemy wartość 0,
- podczas wypisywania wartości liczb na ekran, należy je rzutować na typ integer - (**int**) a[i];,

Rozwiązanie zadania:

1. Pobranie liczby cyfr, dla której będzie realizowana suma,
2. Przydział pamięci (tablica o 1 większa),

```
char *a = new char[N];
char *b = new char[N];
```

3. Pobranie pierwszej liczby,

```
for(int i=0;i<N;i++)
{
        cin >> a[i];
}
//a[0]=0; // zerowanie najstarszej cyfry
```

4. Pobranie drugiej liczby,
5. Utworzenie funkcji ze wstawką asemblerową realizującą sumowanie,

```
void sum_BCD(char * a,char * b, int N)
```

6. Przygotowanie właściwych rejestrów do pętli sumowania

```
mov ecx, N;         //liczba elementów do sumowania
mov edi, a;         //początek 1 tablicy
add edi,ecx;
dec edi;            //na ostatni element tablicy
mov esi, b;         //początek 2 tablicy
add esi,ecx;
dec esi;            //na ostatni element tablicy
```

7. Wyzerowanie rejestru **ax**, oraz flagi CF

```
mov ax, 0;
add al,al          ;//cf=0
```

Etykieta dla pętli sumowania

```
SumBCD:
```

8. Pobranie najmłodszej cyfry liczby **b** (pierwszy obieg pętli)

```
mov al, [esi];          //al = b[N-1];
```

9. Dodanie najmłodszej cyfry liczby **a** + przeniesienie z **ah** (pierwszy obieg pętli)

```
adc al, ds:[edi];       //al = al + [edi] + CF
```

10. Wyzerowanie przeniesienia,

```
mov ah,0;
```

11. Korekta po dodawaniu (ah = ah + dziesiątki), al = jedności,

```
aaa;    //korekta ah = ah + nowe przeniesienie
```

12. Zwrócenie wyniku do **a**,

```
mov ds:[edi], al;       //a[N-1] = al;
```

13. Pozostała część pętli

```
dec esi;
dec edi;
dec ecx;
jnz SumBCD;
```

Uwaga: funkcję **sumaBCD** uruchamiamy dla wartości argumentu rozmiaru (N+1).

2 Iloczyn liczb BCD.

Proszę napisać program wyznaczający iloczyn 2 liczb **BCD** o tej samej liczbie znaków. Liczba znaków ma być pobierana od użytkownika.

Rozwiązanie zadania
W zadaniu tym należy przydzielić pamięć dla liczby **a oraz b** o rozmiarze **N**, dla wyniku – (zmienna **w**) tablicę **(N+N)** elementową. Aby ułatwić proces mnożenia, niezbędna jest dodatkowa tablica **bufor** o rozmiarze **(N+1)**, zawierająca wyniki z pojedynczego mnożenia przez jedną cyfrę. Dla ułatwienia proces pisania, program został podzielony na niezależne etapy, które ułatwią państwu zrozumienie całego programu.

Użycie rejestrów:

- **ecx** - licznik pętli zewnętrznej oraz pętli wewnętrznych,
- **edi** - liczba **b**,
- **esi** - liczba **a**, wewnątrz pętli o etykiecie **loopSuma** rejestr wskazuje na zmienną **bufor**,
- **eax** - rejestr wykorzystywany przy mnożeniu oraz sumowaniu:
  - **ax** - 2 bajtowa młodsza część rejestru **eax**,

  - **al** - młodsza część rejestru (do tej części dodajemy, przez tą część mnożymy, w wyniku korekty przy użyciu instrukcji **aaa** oraz **aam** znajduje się tam młodsza część wyniku (jedności),
  - **ah** - starsza część rejestru - w wyniku korekty przy użyciu instrukcji **aaa** oraz **aam** znajduje się w nim przeniesienie (dziesiątki). **Uwaga** - po operacji sumowania przy użyciu instrukcji **adc**, starsza część rejestru **ah** nie jest zerowana, instrukcja korekty po dodawaniu **aaa** dodaje przeniesienie do wcześniejszej wartość **ah**. Może to być przyczyną błędów przy sumowaniu w przypadku wielokrotnych przeniesień,

- **ebx** - końcowy wynik mnożenia - liczba **wynik**
- **dh** - starsza część rejestru **dx** wykorzystywana do chwilowego zapamiętania przeniesienia podczas sumowania po mnożeniu wartości zdejmowanych ze stosu.

1. Etap 1: wyznaczenie iloczynu dla wszystkich cyfr liczby **a** oraz najmłodszej cyfry liczby **b**, wyznaczenie sumy po przemnożeniu.

   Wyniki z mnożenia dla ułatwienia będziemy przechowywać na stosie. Stos odwraca kolejność danych, tak więc mnożenie musimy wykonywać od najstarszych cyfr do najmłodszych cyfr (odwrotnie w stosunku do mnożenia pisemnego). Po zdjęciu ze stosu, uzyskamy prawidłową kolejność do dalszego sumowania.

   (a) Definicja funkcji wyznaczającej iloczyn BCD

```
void IloczynBCD(char * a, char * b,char *wynik, int N)
```

   (b) Utworzenie dodatkowej zmiennej **bufor**

```
        char *bufor = new char[N+1];
```

   (c) Zabezpieczenie rejestrów procesora

   (d) Ustawienie początkowej wartości rejestrów

```
        mov ecx, N;                 //liczba cyfr
        mov edi, b;
        add edi,ecx;
        dec edi;                    //od najmłodszej
        mov ebx,wynik; // tablica wynikowa o rozmiarze 2N
```

   (e) Wyzerowanie tablicy wyniku mnożenia

```
        add ecx,ecx; //liczba elementów tablicy (N+N)
        add ebx,ecx;
        dec ebx;      //przesunięcie na ostatni element
        xor ax,ax;
zero:
        mov [ebx],al; //wyzeruj element
        dec ebx; //element wcześniej
        Loop zero;
```

   (f) Odtworzenie zawartości rejestrów po zerowaniu

```
        mov ecx,N;
        mov ebx,wynik;
        add ebx,ecx;
        add ebx,ecx;
        dec ebx;          //na koniec tablicy (od najmłodszej)
```

(g) Ustalenie licznika pętli wewnętrznej, ustawienie na początku liczby **a** (na najstarszej cyfrze, realizujemy mnożenie od tyłu),

```
        mov ecx,N;  //pętla  dla zmiennej a [esi]
        mov esi,a;  //początek pierwszej liczby
```

(h) Przygotowanie do mnożenia (tak jak w przypadku sumowania),

```
        mov al,0;
        add al,al          ;//cf=0
```

(i) Pętla mnożenia (od najmłodszej cyfry liczby **b** oraz najstarszej cyfry liczby **a**, dla każdej cyfry liczby **a**, wynik z mnożenia umieszczamy na stosie do późniejszego sumowania w kolejnej pętli,

```
BCDProd_a:      //fragment odpowiedzialny za mnożenie

        mov al, ds:[esi];
        mul ds:[edi];
        aam;    //korekta po mnożeniu, np. 48 => al=8, ah=4,
        push ax; //wynik z mnożenia na stosie [2 bajty]
        inc esi;

Loop BCDProd_a
```

(j) Po wymnożeniu, uzyskuje się na stosie niezsumowane wartości w młodszej części, oraz starszej części, w kolejnym kroku należy te wartości zsumować podczas zdejmowania ze stosu. Ze stosu zdejmujemy od najmłodszej wartości do najstarszej wartości. Sumowanie musimy wykonać ręcznie nie korzystając z instrukcji **adc**. Samemu dodajemy przeniesienie z **ah**. Podczas operacji sumowania też może powstać przeniesienie, tak więc wartość **ah** trzeba gdzieś zapamiętać – np. **dh**.

Przygotowanie do pętli sumowania po przeprowadzonym mnożeniu:

```
        mov ecx,N;
        mov dh,0;   //Na początku przeniesienie 0
        mov esi, bufor; //tablica pomocnicza przechowująca wynik
        // sumowania po mnożeniu
        add esi,ecx; //na koniec tablicy
                     //(wartość najmłodsza)
        mov ax,0;
        add al,al;  //cf=0
```

(k) Pętla sumowania:

```
loopSuma:
        pop ax;          //wynik z mnożenia
        add al,dh;       //przeniesienie z poprzedniej pozycji
        //(starsza część wyniku z poprzedniego obiegu pętli)
        mov dh,ah;       //zapamiętanie przeniesienia z mnożenia
        mov ah,0;
        aaa;             //korekta po sumowaniu, ah=ah+przeniesienie
                         //(nowe przeniesienie w ah)
        add dh,ah;  //dodanie przeniesienia z sumowania (pop. poz.)
        mov [esi],al; // wynik to tablicy bufor
        dec esi;
Loop loopSuma;
```

(l) Pozostaje najstarsze przeniesienie

```
        mov [esi],dh;
```

Po implementacji tej pętli, należy wyświetlić na ekranie wynik sumowania z tablicy **bufor.**

2. Etap 2: Implementacja pętli sumującej wyniki z poszczególnych operacji mnożenia oraz sumowania.

   (a) Pobieramy adres początku bufora, przesuwamy wskaźnik na najmłodszą cyfrę, tablica **bufor** ma rozmiar (N+1), tak więc pętla musi być wykonana (N+1) razy.

```
        mov esi,bufor;
        mov ecx,N;
        add esi,ecx;
        inc ecx;          //jest o 1 element wiecej
        mov ax,0;
        add al,al;
```

   (b) Pętla sumowania bufora oraz tablicy wynikowej o adresie zapisanym w rejestrze **ebx**. Rejestr **ebx** musi być ustawiony w pierwszym obiegu pętli na najmłodszej cyfrze. W kolejnych obiegach pętli zewnętrznej (dla kolejnych cyfr liczby **b**), należy przesuwać rejestr **ebx** na starsze cyfry wyniku o 1 element. Wewnątrz pętli sumowania też zmieniamy położenie tego rejestru. Jego wartość musi być gdzieś zabezpieczona.

```
Suma2:
        mov al,[esi];
        adc al,[ebx];
        mov ah,0;
        aaa;
        mov [ebx],al;
        dec ebx;
        dec esi;
Loop Suma2;
```

3. Etap 3: Implementacja pętli zewnętrznej.

   Etap należy wykonać samodzielnie:

   W pętli zewnętrznej:

   - zmieniamy położenie rejestru **ebx** (wynik mnożenia),
   - zmieniamy położenie rejestru **edi** (cyfry dla liczby **b**),
   - zabezpieczamy wartość rejestru **ebx**, np. stos,
   - zabezpieczamy wartość licznika pętli – rejestr **ecx**.