Katedra Inżynierii Komputerowej
Politechnika Częstochowska

Laboratorium Programowania niskopoziomowego

Laboratorium 5

dr inż. Dziwiński Piotr

15 listopada 2011

1 Suma elementów macierzy

Proszę napisać program wyznaczający sumę elementów macierzy. Elementy macierzy oraz rozmiary należy pobrać od użytkownika.

Fragment programu realizujący to zadanie w języku C++ może wyglądać następująco:

```
int suma_matrix_1(int ** macierz, int n1, int n2)
{
        int suma=0;
        for(int i=0;i<n1;i++)
                for(int j=0;j<n2;j++)
                {
                        suma+=macierz[i][j];
                }
                return suma;
}
```

Wygląda on dla wszystkich bardzo znajomo. Raczej wszyscy jesteśmy przyzwyczajeni do wykorzystywania pętli **for**. Bardziej efektywnym rozwiązaniem będzie przybliżenie kodu C++ do rozwiązania bliższego asemblerowi – wykorzystajmy do tego celu pętlę **do** ... **while**(...)

```
int suma_matrix_2(int ** macierz, int n1, int n2)
{
        int suma=0;
        int *m2;
        do
        {
                m2 = macierz[--n1];
                do
                {
                        suma+=m2[--n2];
                } while(n2>0);
        } while(n1>0);
        return suma;
}
```

Rozwiązanie to pokrywa się ze sposobem organizowania obiegu pętli w asemblerze. Wzorując się na typ przykładzie można przystąpić do implementacji zadania w asemblerze.

Rozwiązanie zadania

1. Zabezpieczenie rejestrów procesora
2. Ustawienie stanu początkowego dla wybranych rejestrów

```
        mov esi,macierz;
        mov ecx, n1;      //i
        mov ebx, n2;      //j
        xor eax,eax;      //suma
        //dodatkowo edi <- m2 z przykładu wcześniej
```

3. Utworzenie pierwszej pętli (zewnętrznej), indeks **i lub n1** -> **ecx**

```
petla1:

        loop petla1;
```

4. Pobranie adresu wiersza do rejestru **edi** (wewnątrz pętli)

```
mov edi,[esi + 4 *ecx -4]; //macierz[i] (i - ecx)
```

5. Przygotowanie rejestru do indeksowania kolumn macierzy dla pętli wewnętrznej

```
mov ebx, n2; //j
```

6. Utworzenie drugiej pętli (wewnętrznej), indeks **j lub n2** -> **ebx**

```
petla2:

        dec ebx;
        jnz petla2;
```

7. Sumowanie elementów w pętli wewnętrznej

```
add eax,[edi + 4 *ebx -4]; //macierz[i][j] (i - ecx,j - ebx)
```

8. Zwrócenie wyników

9. Przywrócenie stanu rejestrów procesora

2 Mnożenie macierzy i wektora

Proszę napisać metodę mnożącą macierz $\mathbf{M}_{N\times K}$ i wektor $\mathbf{V}_K$ zgodnie z następującym wzorem:

$$u_n = \sum_{k=0}^{K-1} m_{nk} \cdot v_k \tag{1}$$

gdzie: N - liczba wierszy macierzy $\mathbf{M}$, K - liczba kolumn macierzy $\mathbf{M}$.

Operację mnożenia należy zaimplementować w C++ oraz w asemblerze.

Rozwiązanie (implementacja C++, wskaźniki):

```
int * iloczyn(int **m, int *v,int *u,int N, int K)
{
        for(int n=0;n<N;n++)
        {
                u[n]=0;
                for(int k=0;k<K;k++)
                {
                        u[n] = u[n] + m[n][k] * v[k];
                }
        }
        return u;
}
```

Rozwiązanie (implementacja asembler, wskaźniki):

Opis zastosowania rejestrów:

- **edi** - adres początku wektora **v**,

- **ebx** - adres początku **u** (1 funkcja) oraz wartość elementów wektora sumowanych w pętli wewnętrznej (2 funkcja), podczas sumowania adres początku wektora jest zabezpieczony na stosie,
- **ecx** - rejestr wykorzystywany jako licznik pętli zewnętrznej (1 funkcja) oraz wewnętrznej (2 funkcja), przed wejściem w pętlę wewnętrzną jest zabezpieczony na stosie,
- **esi** - adres początku macierzy (poruszanie po wierszach), rejestr jest również używany do przemieszczania po kolumnach, w każdej pętli zewnętrznej pobierana jest ponownie wartość adresu początku macierzy, wartość tą można również zabezpieczyć na stosie,
- **eax** - rejestr wykorzystywany do operacji mnożenia składników wektora oraz macierzy, używany także jako chwilowy bufor dla wartości z rejestru **ebx** z pętli wewnętrznej,

1. Definicja funkcji

```
int * iloczyn_asm(int **m, int *v,int *u,int N, int K)
```

2. Zabezpieczenie rejestrów procesora
3. Przygotowanie rejestrów procesora przed pętlą zewnętrzną

```
        mov edi,v;
        mov ebx,u;
        mov ecx,N; //pętla zewnętrzna (od końca)
```

4. Etykieta pętli zewnętrznej

```
for11:
```

5. Przejście do ostatniego wiersza macierzy m[N−1]

```
        mov esi,m;
        mov esi,[esi + 4*ecx -4]; //ostatni wiersz macierzy
```

6. Zabezpieczenie rejestru **ecx** oraz **ebx** na stosie, rejestry spełniają drugie funkcje wewnątrz pętli wewnętrznej,

```
        push ecx;
        push ebx;
```

7. Przygotowanie rejestru **ebx** do sumowania, wprowadzenie drugiego licznika pętli wewnętrznej do rejestru **ecx**,

```
        mov ebx,0;
        mov ecx, K; //pętla wewnętrzna
```

8. Etykieta pętli wewnętrznej

```
for22:
```

9. Pobranie elementu z macierzy m, pierwsze wywołanie m[N-1][K-1], mnożenie przez ostatni element wektora v, dodanie wartości do rejestru **ebx**

```
        mov eax, [esi + 4*ecx -4] //m[n][k]
        mul [edi + 4*ecx -4];
        add ebx, eax;
```

10. Zakończenie pętli wewnętrznej

```
loop for22;
```

11. Zwrócenie wyników sumowania do rejestru **eax**, przywrócenie wartości dla (funkcji 1) rejestrów procesora **ebx**,**ecx**,

```
mov eax, ebx;  //wynik wewnętrznej pętli
pop ebx;
pop ecx;
```

12. Zapisanie wyniku sumowania do wektora **u**,

```
mov [ebx +4*ecx - 4],eax;
```

13. Zakończenie pętli zewnętrznej

```
loop for11;
```

14. Zwrócenie adresu wektora wynikowego.