

Procesor sygnałowy ADSP21161.

Przykładowe programy do ćwiczenia: Projekt: **Podstawowe operacje**
(*Project>Open> Podstawowe operacje.dpj*)

Pętle programowe i sprzętowe

1. Napisać pętlę programową realizującą algorytm $y=n!$ (*silnia*) w języku C:

```
int i,n;  
float y;  
n=15;  
y=1;  
for (i=1;i<=n; i++)  
    y*=i;
```

- a. Przeanalizować działanie ww. procedury w pracy krokowej. Wykonać program w pracy krokowej (klawisz **F11**) przy otwartym oknie *View>Debug Windows>Disassembly*. Zapisać sposób realizacji ww. procedury przez kompilator w języku niskiego poziomu (w assemblerze).
- b. Zarejestrować czas działania ww. procedury w cyklach korzystając z następującego sposobu:

```
int time_start, time_elapsed;  
....  
time_start=count_start();  
... procedura, której czas działania chcemy zmierzyć ...  
time_elapsed=count_end(time_start);  
printf("Czas obliczen %d ",time_elapsed);
```

2. Przeanalizować działanie podobnej procedury *silnia_ASM* napisanej w języku niskiego poziomu (assembler). Opisać różnice w stosunku do procedury w języku C. Zarejestrować czas działania ww. procedury w cyklach.
3. Sprawdzić działanie ww. procedury dla wartości $n=20$. Skomentować wyniki. Opisać możliwość rozwiązania zaistniałego problemu.

Macierze i wektory

4. Przeanalizować procedurę (*matrix_x_vector_1*) mnożenia macierzy (**matrix_A**) przez wektor (**vector_X**) zawartą w przykładzie nr 1. Wykonać program w pracy krokowej (klawisz **F11**) przy otwartym oknie *View>Debug Windows>Disassembly*.
 - a. Zaobserwować działanie skoku opóźnionego *cjump_matrix_x_vector_1(db)* przy wejściu do procedury *matrix_x_vector_1*
 - b. Zaobserwować działanie dwóch pętli zrealizowanych sprzętowo (*loop_kolumny, loop_wiersze*).
 - c. Zanotować czas wykonywania się ww. procedury w cyklach
 - d. Napisać zmodyfikowaną wersję ww. procedury bez wykorzystania pętli – dla ustalonego rozmiaru macierzy $n=3$. Zanotować czas wykonywania się ww. procedury w cyklach.
 - e. Porównać działanie procedury (*matrix_x_vector_2*) wykorzystującą operacje wykonywane równoległe (odczyt równoczesny z obu pamięci **PM** i **DM**). Zanotować czas wykonywania się ww. procedury w cyklach. Uwzględnić (odjąć od obu czasów) czas wejścia/wyjścia do procedury (Mierzając czas wykonania pustej procedury *matrix_x_vector_0*).
5. Napisać procedurę mnożenia *macierz x macierz* wykorzystując bufory kołowe, oraz operacje równoległe.
6. Napisać procedurę mnożenia *macierz x macierz* bez wykorzystania najbardziej wewnętrznej pętli. Porównać czasy wykonywania obu procedur.

Podstawowe algorytmy

1. Suma wartości bezwzględnych elementów tablicy
- 2.2. wyszukiwanie MIN, MAX w tablicy
- 3.3. mnożenie macierzy: macierz x wektor
- 4.4. suma ważona wejść
(model neuronu z liniową funkcją aktywacji)

-Suma wartości bezwzględnych elementów tablicy

```
I0 = tablica; //adres tablicy
M0 = 1;      //zwiększanie adresu o 1
L0 = 0;      //nie ma bufora kołowego
R0 = 0;      //wartość początkowa sumy
LCNTR = 30, DO the_end UNTIL LCE; //Pętla powtarzana 30 razy
R1 = DM(I0,M0);
R1 = ABS(R1);
the_end: R0 = R0 + R1; // ostatnia instrukcja w pętli
```

-Suma wartości bezwzględnych 30 elementów tablicy wariant II

```
I0 = tablica; //adres tablicy
M0 = 1;      //zwiększanie adresu o 1
L0 = 0;      //nie ma bufora kołowego
R0 = 0;      //wartość początkowa sumy
R1 = DM(I0,M0); //pierwsza dana
LCNTR = 30-1, DO the_end UNTIL LCE; //Pętla powtarzana 29 razy
R2 = ABS(R1), R1 = DM(I0,M0);
nop; // instrukcja dodatkowa !!! (skok opóźniony)
the_end: R0 = R0 + R2; // ostatnia instrukcja w pętli
R2 = ABS(R1); // instrukcje wykonywane jeden raz R0 = R0 + R2;
```

-Suma wartości bezwzględnych 30 elementów tablicy wariant III

```
I0 = tablica; //adres tablicy
M0 = 1;      //zwiększanie adresu o 1
L0 = 0;      //nie ma bufora kołowego
R0 = 0;      //wartość początkowa sumy
R1 = DM(I0,M0); //pierwsza dana
R2 = ABS(R1), R1 = DM(I0,M0); //druga dana R0 = R0 + R2;
LCNTR = 15-1, DO the_end UNTIL LCE; //Pętla powtarzana 14 razy
R2 = ABS(R1), R1 = DM(I0,M0);
R0 = R0 + R2;
R2 = ABS(R1), R1 = DM(I0,M0);
the_end: R0 = R0 + R2; // ostatnia instrukcja w pętli
R2 = ABS(R1); // instrukcje wykonywane jeden raz R0 = R0 + R2;
```

-Wyszukiwanie MIN, MAX w tablicy

```
vector_maximum: bit set MODE1 RND32 | PEYEN;

b0 = INPUT; //address of vector INPUT
i0 = b0;
l0 = 0;
m0 = 1;

f4=0.0;
f0=dm(i0,m0);
lcntr=N-1, do Vecmax until lce; /* vector maximum loop */
f1 = dm(i0,m0);
f0 = max( f0, f1);
nop;
Vecmax:
rts (db);
bit clr MODE1 PEYEN;
nop;
```

-Wyszukiwanie MIN,MAX w tablicy - wersja II

```
vector_maximum:    bit set MODE1 RND32 | PEYEN;

    b0 = INPUT;    //address of vector INPUT
    i0 = b0;
    l0 = 0

    f4=0.0;
    f0=dm(i0,2);
    lcntr=r1, do vecmax until lce;    /* vector maximum loop */
        comp(f4,f0), f8=dm(i0,2);    // last read is appended offset for vector location */
        if lt f4=pass f0, r2=i0;
        comp(f4,f8), f0=dm(i0,2);
vecmax:    if lt f4=pass f8, r2=i0;
    r2=r2-r0, r0=b0;
    r2=r2-r0, r0=s4;
    comp(f4,f0);
    if lt f4=pass f0, r2<->s2;
    rts (db);
    bit clr MODE1 PEYEN;
    nop;
vector_maximum.end;
```

-Mnożenie macierzy

macierz x wektor: $Y = A \times B$

```
I0 = MA;    //adres początku macierzy A
M0 = 1;    //zwiększanie adresu o 1 po każdym odczycie
L0 = 0;    //nie ma bufora kołowego
I1 = MB;    //adres początku wektora B
M1 = 1;    //zwiększanie adresu o 1 po każdym odczycie
L1 = N;    //bufor kołowy o rozmiarze N
B1 = MB;
I3 = MY;    //adres początku wektora Y
M3 = 1;    //zwiększanie adresu o 1 po każdym odczycie
L3 = 0;    //nie ma bufora kołowego
LCNTR = N, Do wiersze UNTIL LCE;
MRF = 0;    //wartość początkowa sumy
LCNTR = N, DO kolumny UNTIL LCE;    R1 = DM(I0,M0);
R2 = DM(I1,M1);
kolumny: MRF = MRF +R1 * R2; // ostatnia instrukcja w pętli wewn.
DM(I3,0) = MR1; //zapisanie wyniku (kolejnego wiersza)
nop;    //instrukcja dodatkowa
Wiersze: nop;
```

-Suma ważona wejść

(model neuronu z liniową funkcją aktywacji)

```
I0 = WX;    //adres początku wektora wejściowego WX
M0 = 1;    //zwiększanie adresu o 1 po każdym odczycie
L0 = N;    //bufor kołowy (rozmiar wektora wejść)
B0 = WX;
I1 = MB;    //adres początku warstwy wyjściowej neuronów
M1 = 1;    //zwiększanie adresu o 1 po każdym zapisie
L1 = 0;    //bez bufora kołowego    I2 = MY;    //adres początku wag
neuronu
M2 = 1;    //zwiększanie adresu o 1 po każdym odczycie
L2 = 0;    //nie ma bufora kołowego
R0 = DM(I2,M2);    //waga zerowa
MRF = R0;    //wartość początkowa sumy
LCNTR = N, Do wejścia UNTIL LCE;
R1 = DM(I0,M0); //kolejne wejście
R2 = DM(I2,M2); //i odpowiednia waga
wejścia: MRF = MRF +R1 * R2; // ostatnia instrukcja w pętli wewn.
DM(I1,M1) = MR1; //zapisanie wyniku (suma ważona neuronu)
```

-Suma ważona wejść

- wersja zmiennoprzecinkowa

```
I0 = WX;    //adres początku wektora wejściowego WX
                (pamięć DM)
M0 = 1;    //zwiększanie adresu o 1 po każdym odczycie
L0 = N;    //bufor kołowy (rozmiar wektora wejść)
B0 = WX;
I1 = MB;    //adres początku warstwy wyjściowej neuronów
                (pamięć DM)
```

```

        M1 = 1;          //zwiększanie adresu o 1 po każdym zapisie
        L1 = 0;          //bez bufora kołowego          I2 = MY;          //adres początku wag
neuronu (pamięć PM)
        M2 = 1;          //zwiększanie adresu o 1 po każdym odczycie
        L2 = 0;          //nie ma bufora kołowego
        F0 = DM(I2,M2);   //waga zerowa =wartość początkowa sumy
        LCNTR = N, Do wejścia UNTIL LCE;
        F1 = DM(I0,M0); F2 = PM(I2,M2); //kolejne wejście i waga          F3 = F1 * F2;
wejścia: F0 = F0 + F3; // ostatnia instrukcja w pętli wewn.
        DM(I1,M1) = F0;   //zapisanie wyniku (suma ważona neuronu)

```