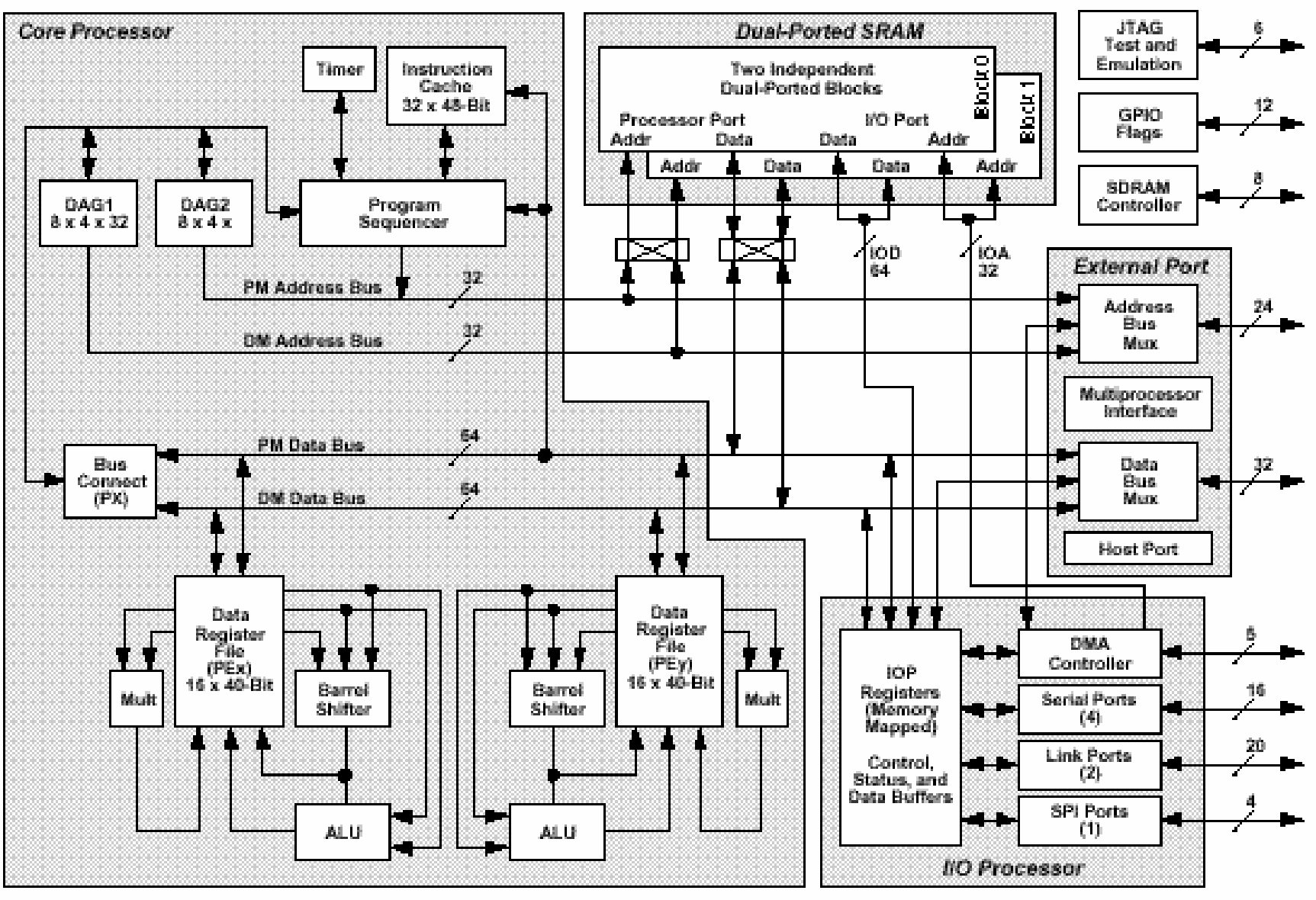
# Architektura SIMD i model programowania

**Sekcja 8**

# ADSP-21161: Diagram blokowy architektury



# SIMD ADSP-21160 pojecia

**SISD = Single Instruction Single Data (pojedyncza instrukcja, pojedyncza dana)**

– uklady z rodziny ADSP-2106x SHARC

**SIMD = Single Instruction Multiple Data ( pojedyncza instrukcja, wielo raka dana)**

– W trybie SIMD , identyczne instrukcje wykonane na jednostkach PEx i PEy; róznica to dana

- **Element przetwarzajacy = ALU, Multiplier, Shifter, & Register file**
- **Operacje Explicit(jawne) vs. Implicit(niejawne)**
  - Operacje Explicit sa pisane w kodzie zródlowym i wykonywane w PEx
  - Operacje Implicit sa wykonywane w PEy. Sa nazywane implicit poniewaz zapewniaja te same operacje jak w PEx i NIE sa domyslnie pisane w kodzie zrodlowym
    - w przykladowej instrukcji pokazanej tutaj , r0 = r1+r2; jest operacja explicit dla PEx, operacja implicit , która wykonuje sie w PEy (w tym samym cykl) jest s0=s1+s2. Rejestry r0, r1, i r2 sa formalnie zdefiniowanymi rejestrami PEx a rejestrami implicit w PEy sa s0, s1, i s2

# Cechy SIMD ADSP-21161

- **SIMD jest TRYBEM operacji**
- **Rozwój dla architektury 2106x**
- **Kazda instrukcja obliczeniowa lub z dostepem dodanych jest wykonywana w obu elementach przetwarzajacych**
- **Kazdy z elementow przetw. operuje na roznych danych**
- **Zwiekszenie szerokosci szyn danych PM i DM do 64bitow**
  - umozliwia przeplyw czterech 32-bitowych slow w kazdym cyklu (dwóch przez kazda z szyn danych)

# SIMD Mode Enable Bit (bit aktywujacy tryb SIMD)

Zalaczajac plik def21161.h, uzyj nastepujacej skladni :

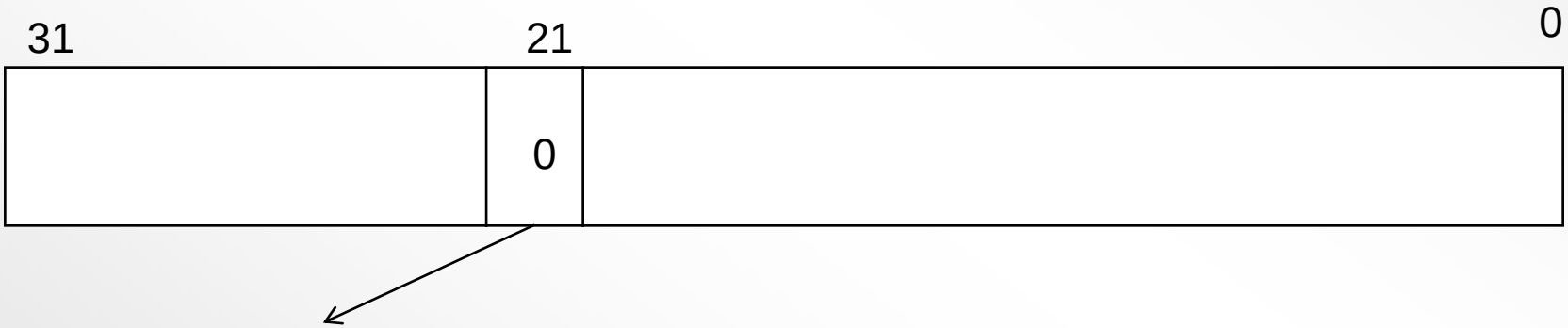
**bit set mode1 PEYEN;**

**bit clr mode1 PEYEN;**

Symboliczne nazwy makr “PEYEN” and “SIMD”sa zdefiniowane by korespondowac z MODE1 Bit 21 , moga byc uzyte zeby uruchomic tryb SIMD

Uwaga: rejestr MODE1 ma jednocyklowe opoznienie (zmiany daja efekt w drugim cyklu po zmianie

MODE1 rejestr




PEYEN = Enable dla przetwarzanego elementu Y lub SIMD Enable Bit

0 ==> tylko PEx aktywowane(enabled)

1 ==> PEx i PEy aktywowane(enabled)

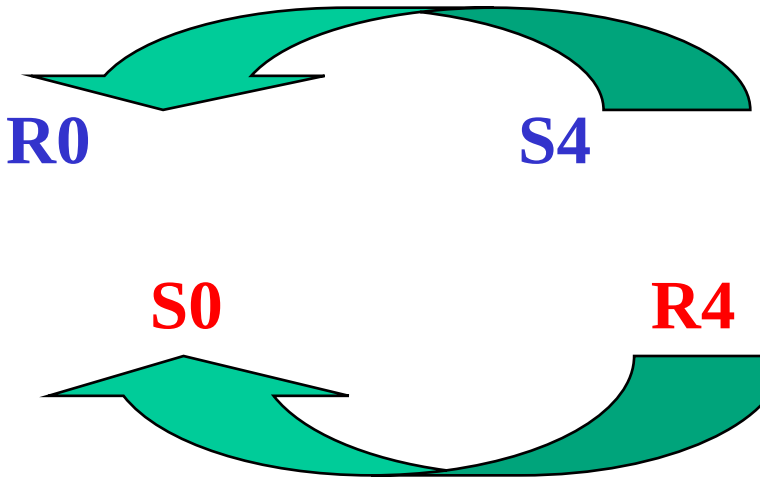
# Obliczenia i operacje z dostepem do danych: SISD vs SIMD

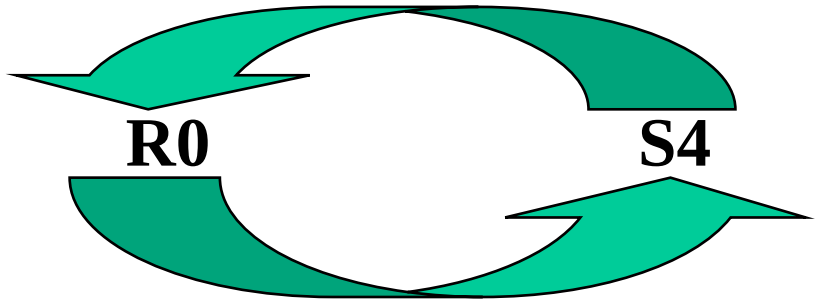
- **SIMD Enabled(aktywowany tryb SIMD):**
  - –kiedy jakiekolwiek obliczenie lub dostep do danych ma miejsce, bedzie wykonywac sie automatycznie w obu elementach przetwarzajacych
    - F0 = F1 + F2; formalnie zdefiniowane w kodzie programu
      - –bedzie wykonane w PEx
    - S0 = S1 + S2 operacja niejawna nie zdefiniowana w kodzie zrodlowym
      - –bedzie automatycznie wykonana w PEy w tym samym cyklu instrukcji
- **SIMD Disabled (aktywowany tryb SISD):**
  - –tylko instrukcje jawne beda wykonane w PEx

# Transfer i wymiana danych miedzy rejestrami
## SISD vs SIMD

- **R0 = S4;**
- **SISD: transfer ma miejsce z jednego rejestru do innego**
- **SIMD: transfer pomiedzy R0 and S4 w PExi podobnie transfer pomiedzy rejestrami S0 i R4 w PEy**

- **R0 <-> S4;**
- **Wymiana pomiedzy rejestrem danych w PEx i rejestrem danych w PEy**
- **Opercja wymiany jest taka sama w SISD jak w SIMD; zadna dodatkowa operacja nie ma miejsca w trybie SIMD .**



# ADSP-21161 Dostep do danych

# ADSP-21161: parametry dostepu do pamieci

Opcje, które wplywaja na liczbe bitów dostepnych z pamieci:

- **SINGLE(pojedynczy) lub DUAL ACCESS(podwojny dostep)**
  - –pojedynczy dostep wyglada tak: r0= DM(i0,m0);
  - – podwojny dostep wyglada tak: r0= DM(i0,m0), r4= PM(i8,m8);
- **ADDRESS SPACE(przestrzen adresowa)**
  - – Short Word, Normal Word, Long Word
    - –opcja instrukcji LW (Long-Word)
      - –zapewnia 64bitowy dostep w przestrzeni Normal Word
- **Tryb SISD VS. SIMD (aktywowany przez bit w MODE1)**
- **Tryb BROADCAST (aktywowany przez bit w MODE1)**
- **EXTENDED PRECISION VS. NORMAL PRECISION**

# Dostep do pamieci: rozmiar dostepnych słów

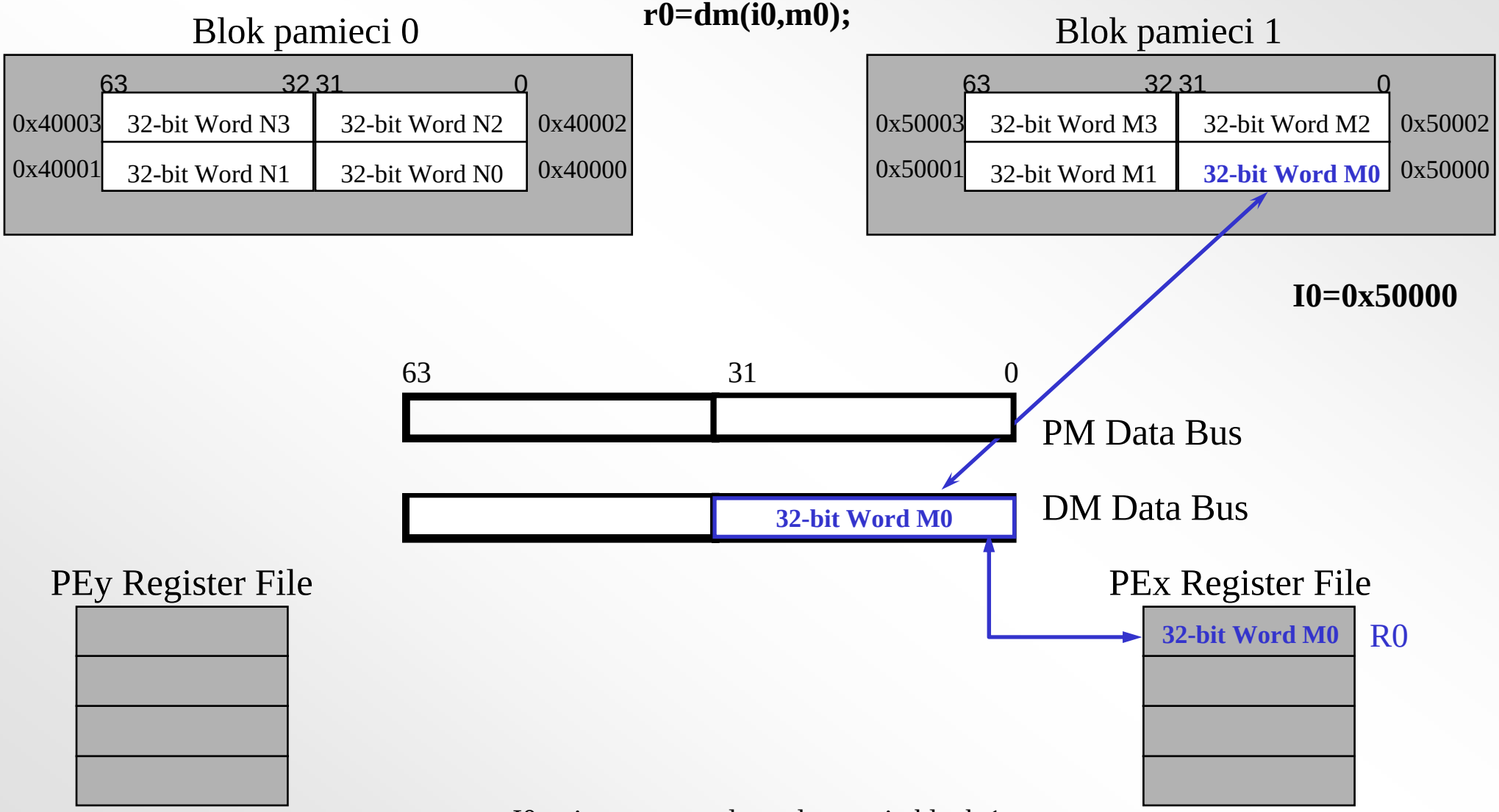
- **Rozmiar dostepnych  słów**
  - Long Word (dwa slowa 32-bitowe)
  - Extended Precision Normal Word (slowo 40-bitowe)
  - Normal Word (slowo 32-bitowe)
  - Short Word (slowo16-bitowe)
- **Przestrzen adresowa**
  - Long word space: 64 bity/dostep
  - Normal word space: 32 bity/dostep
  - Short word space: 16 bity/dostep

# Dostep do pamieci: Tryb DSP SISD czy SIMD

- **Tryb SISD VS. SIMD(uruchamiany przez bit w MODE1)**
  - SISD implikuje, ze dane pojda tylko do jednego register file
  - SIMD implikuje, ze dane pojda do obu rejestrow PEx i PEy
  - Parowanie rejestrów: r0/s0, r1/s1, ……, r15/s15 (“complimentary” registers, i.e. r0 i s0)
  - Relacja jawna pomiedzy rejestrami PEx i PEy koresponduje do komplementarnych par rejestrow
  - Transfery SIMD maja zawsze wynik w transferze danych pomiedzy rejestrem jawnym i adresem jawnym i pomiedzy rejestrem niejawnym i adresem niejawnym

# Normal-Word Space - Single Data - SISD

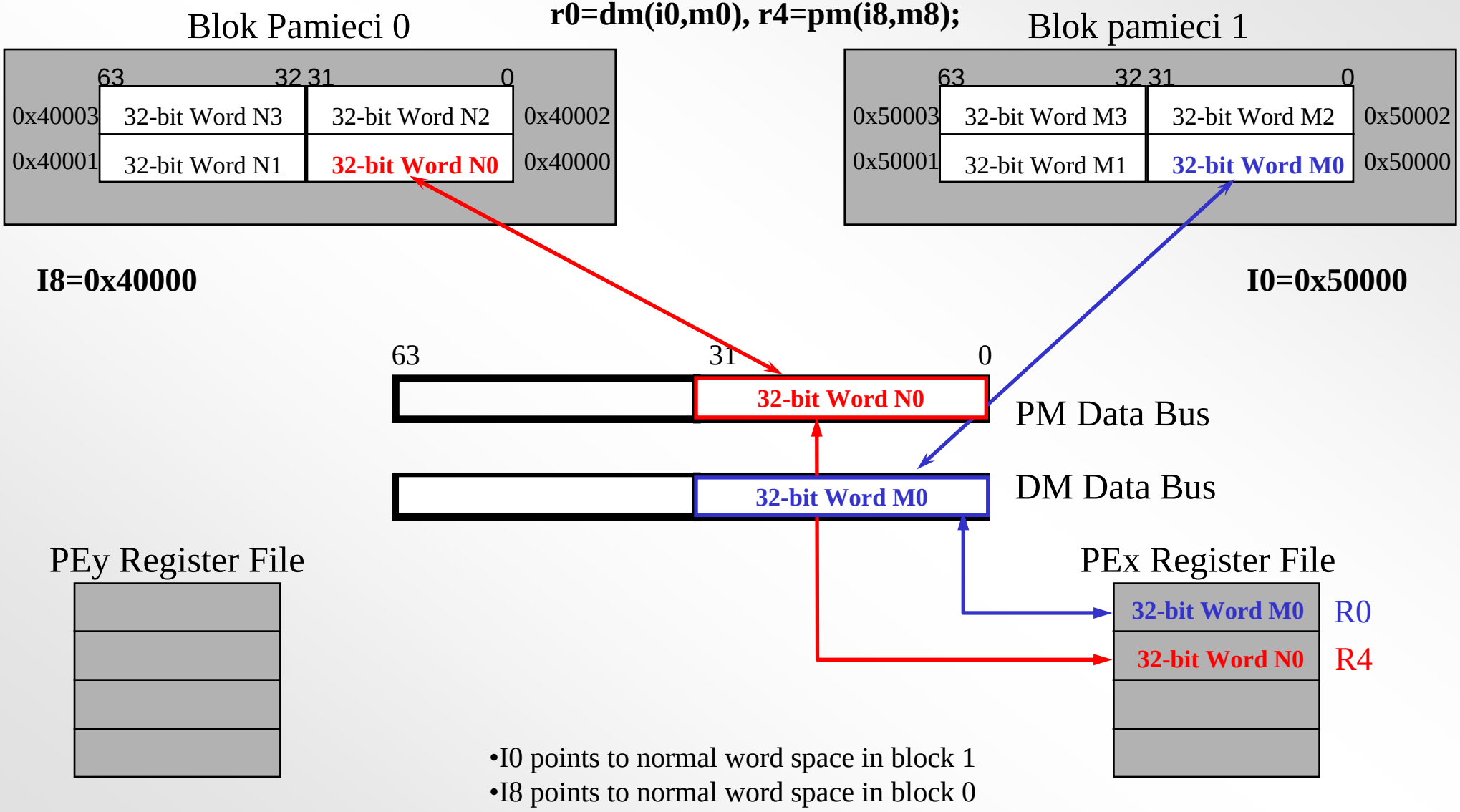
**Przyklad:**
**r0=dm(i0,m0);**




•I0 points to normal word space in block 1

# Normal-Word Space - Dual Data - SISD

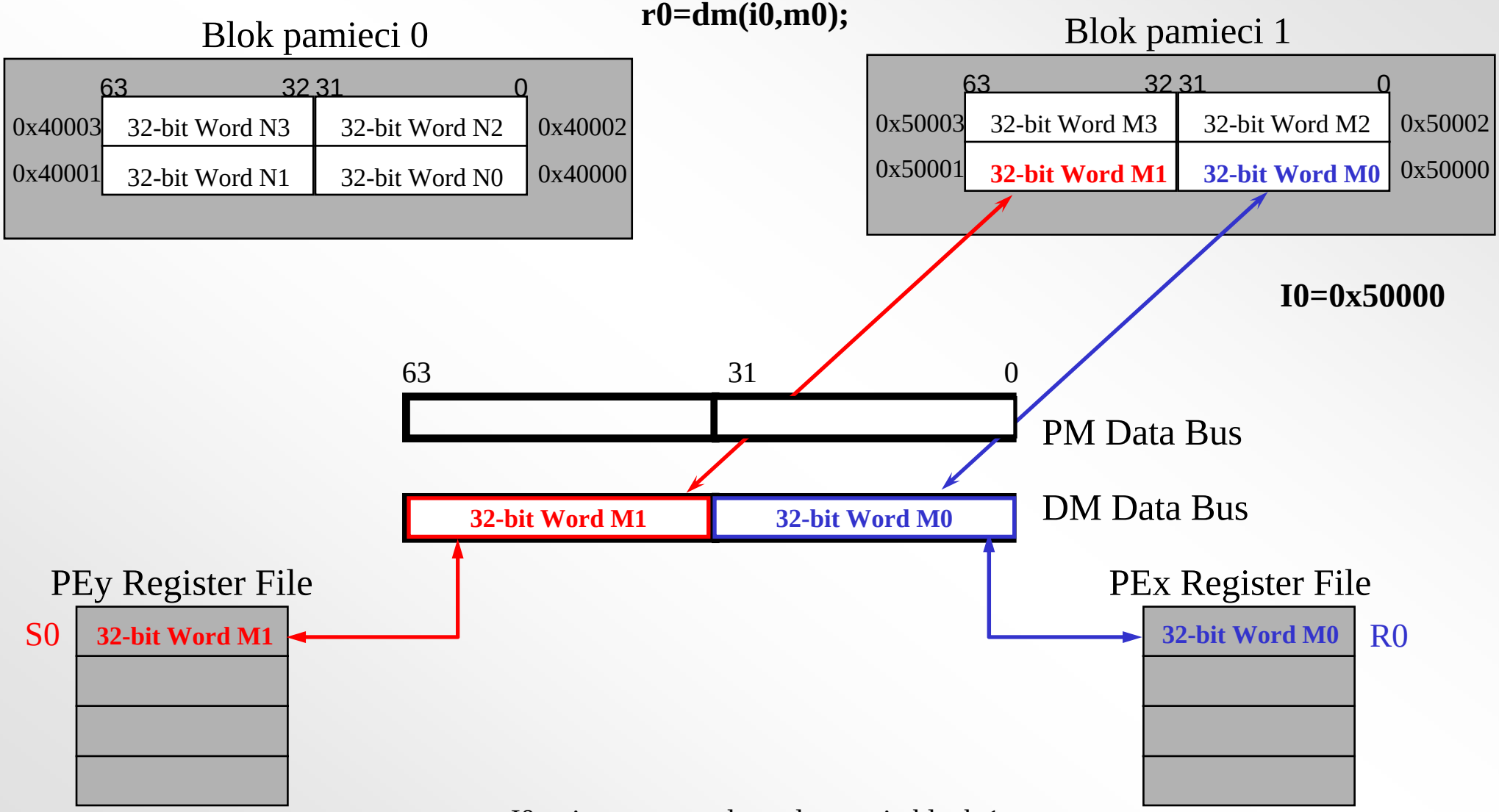
**Przyklad:**
**r0=dm(i0,m0), r4=pm(i8,m8);**




- I0 points to normal word space in block 1
- I8 points to normal word space in block 0

# Normal-Word Space - Single Data - SIMD

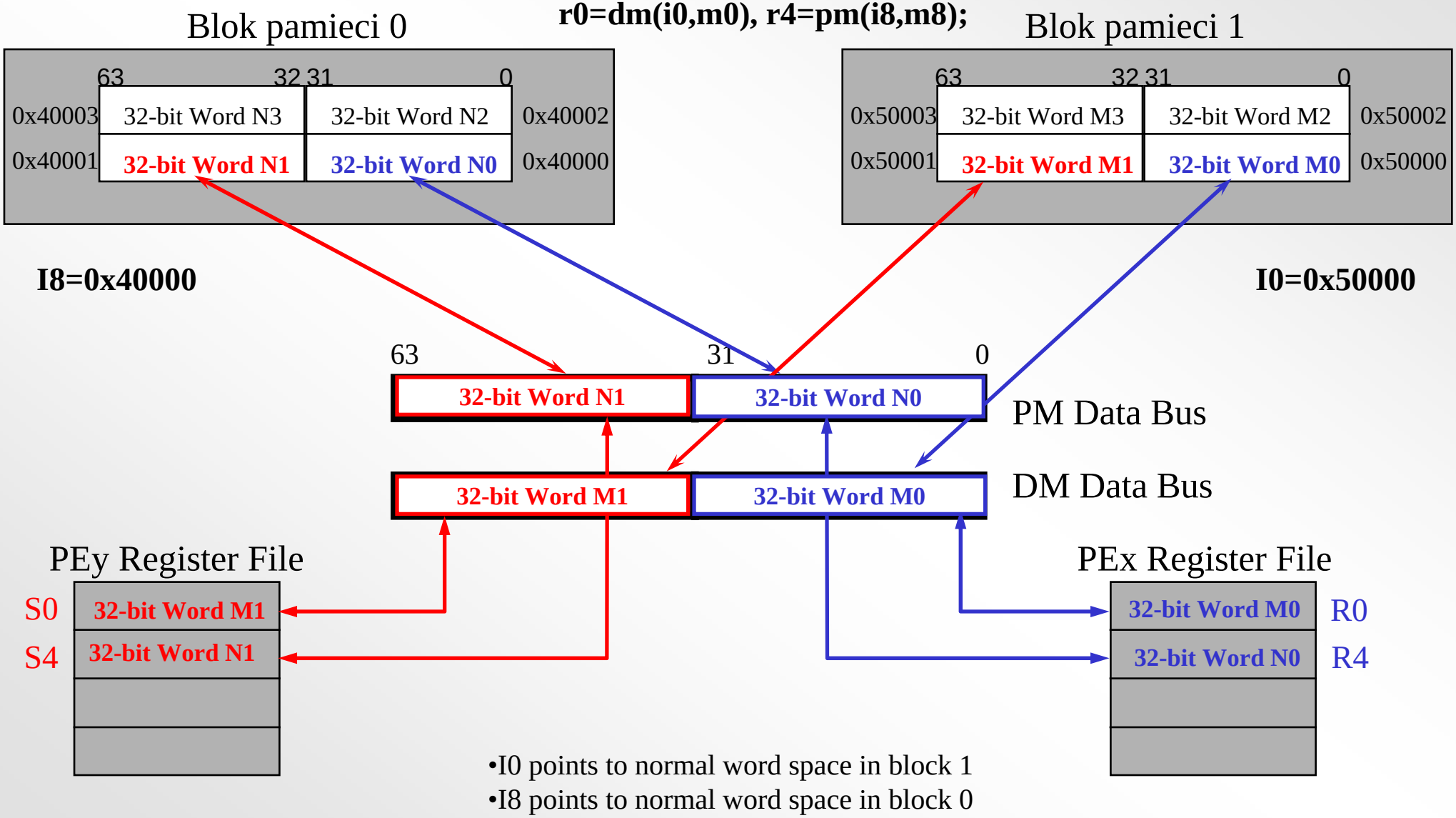
**Przyklad:**
**r0=dm(i0,m0);**




•I0 points to normal word space in block 1

# Normal-Word Space - Dual Data - SIMD

**Przyklad:**
**r0=dm(i0,m0), r4=pm(i8,m8);**




•I0 points to normal word space in block 1
•I8 points to normal word space in block 0

# ADSP-21161 dostep do pamieci: Tryb Broadcast

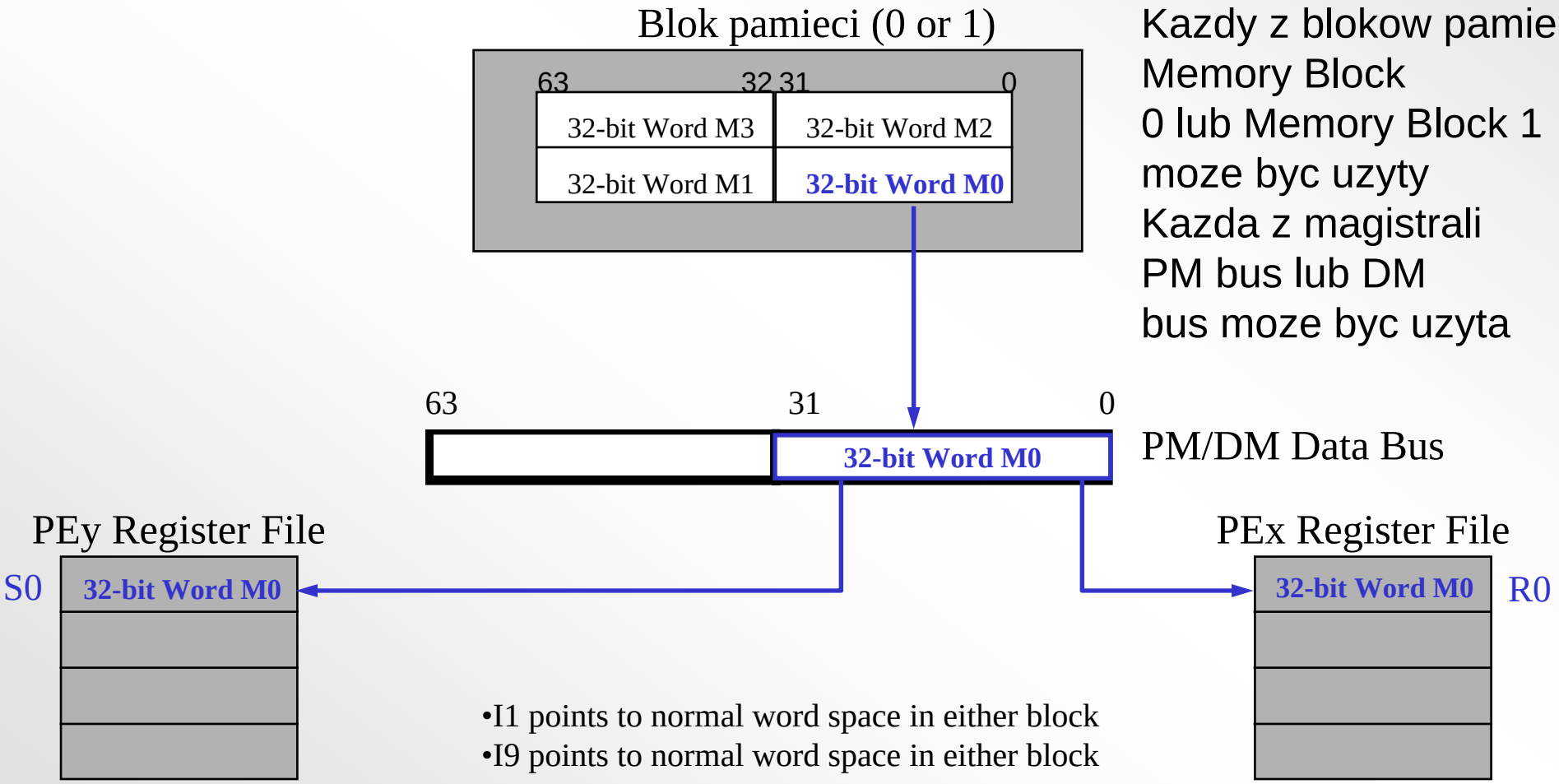
- **Tryb BROADCAST(uruchamiany przez bit w MODE1)**
  - –Tryb Broadcast Mode pracuje tak samo w SISD i SIMD. Ten tryb jest niezalezny od bitu SIMD enable w rejestrze MODE1
  - –Tylko do odczyów pamieci
  - – Dana z jednej lokacji pamieci przeplywa do obu rejestrow Register Files
  - – Wyzwalanie i uzycie trybu Broadcast
    - –Jeden rejestr w kazdym DAG uzywany dla Broadcast (I1, I9)
    - – BDCST1 zezwala na broadcasting dla i1 z DAG1
    - – BDCST9 zezwala na broadcasting dla i9 z DAG2

# Normal-Word Space - Single Data - Broadcast

**r0=dm(i1,m1);**

***-OR-***

**r0=pm(i9,m9);**

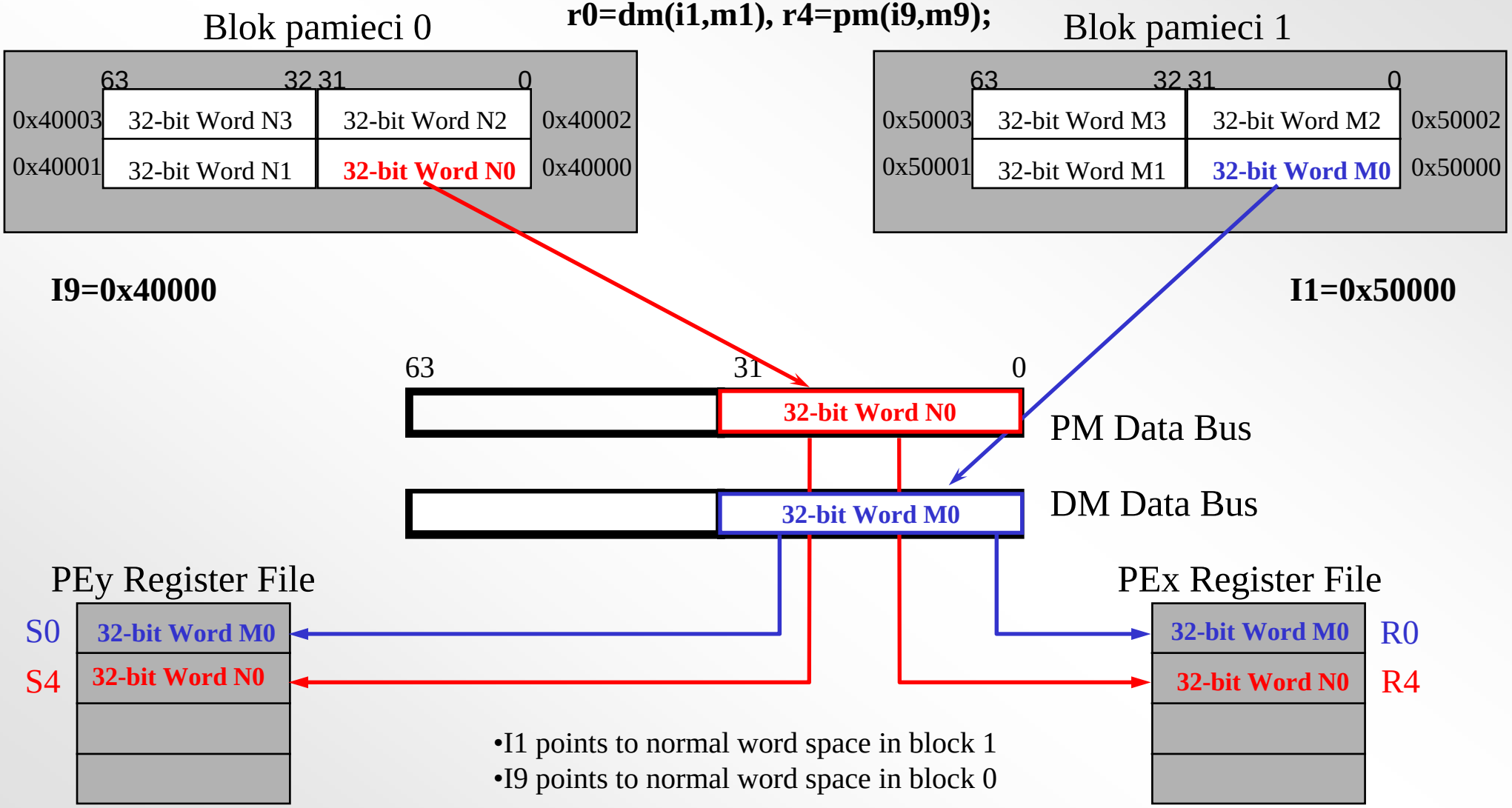



Kazdy z blokow pamieci:
Memory Block
0 lub Memory Block 1
moze byc uzyty
Kazda z magistrali
PM bus lub DM
bus moze byc uzyta

- I1 points to normal word space in either block
- I9 points to normal word space in either block

# Normal-Word Space - Dual Data - Broadcast

**Przyklad:**
**r0=dm(i1,m1), r4=pm(i9,m9);**




- I1 points to normal word space in block 1
- I9 points to normal word space in block 0

# ADSP-21161 Dostep do pamieci: Long Word

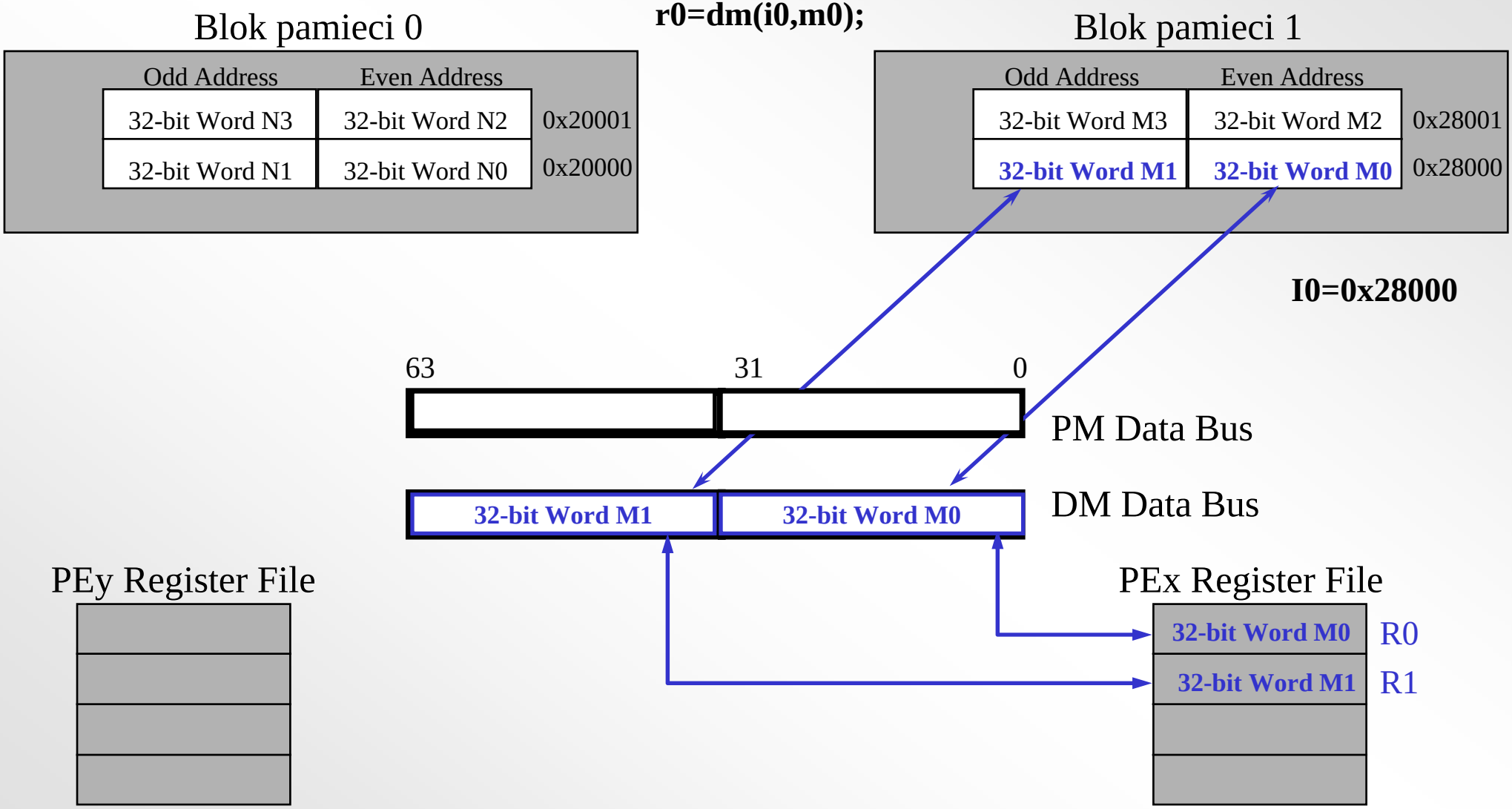
- **Adresowanie Long Word**
  - Dla słów dostepu do przestrzeni adresow Long word (lub opcji instrukcji Long word) 64 bity sa dostepne dla/z sasiednich rejestrów w tym samym register file (i.e. r0/r1)
  - Parowanie rejestrów: r0/r1, r2/r3, .., r14/r15 (rejestry “neighbor” zawsze zaczynaja sie z parzystym rejestrem , i.e. r0 i r1 **NIE** r1 i r2)
  - Transfery Long word [przestrzen adresowa Long word lub ustawiona LW ] zawsze daja wynik w transferze danych pomiedzy rejestrem jawnym i 32 LSBs i pomiedzy rejestrem (neighbor) niejawnym i 32 MSBs

# ADSP-1161 Dostep do pamieci: Long Word

- **Opcje instrucji Long-Word (LW)**
  - Po przetransponowaniu adresu, ten dostep ustala typ funkcji dokladnie jak przestrzen adresowania Long word
  - Dotyczy tylko dostepu do danych w przestrzeni normal-word ,wyglada jak: r0 = dm(i0,m0) (lw);
  - Opcje long-word  sa wazniejsze niz pozostale czynniki (SIMD, IMDWx)

# Long Word Space - Single Data - SISD

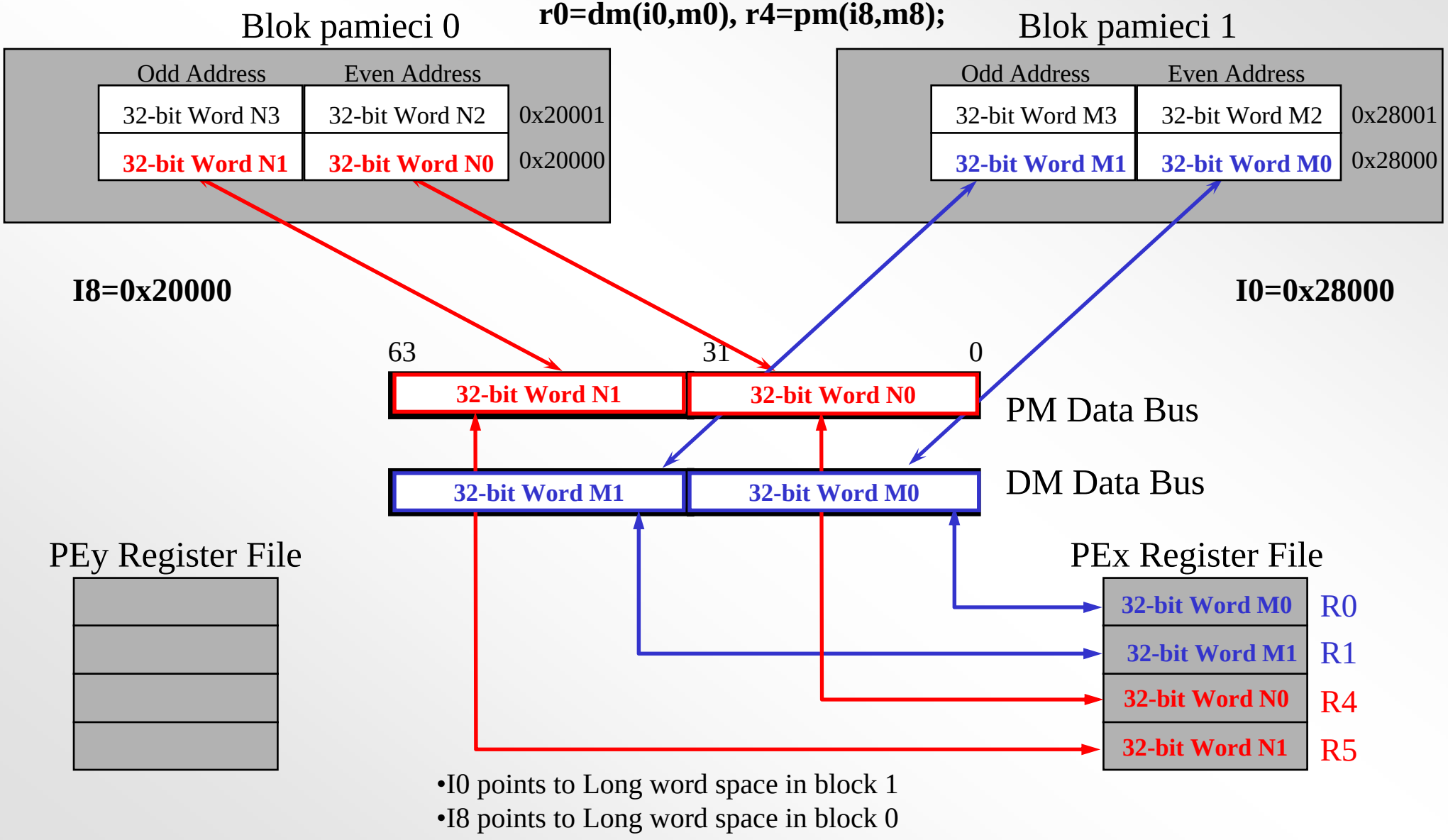
**Przyklad:**
**r0=dm(i0,m0);**




•I0 points to Long word space in block 1

# Long Word Space - Dual Data - SISD

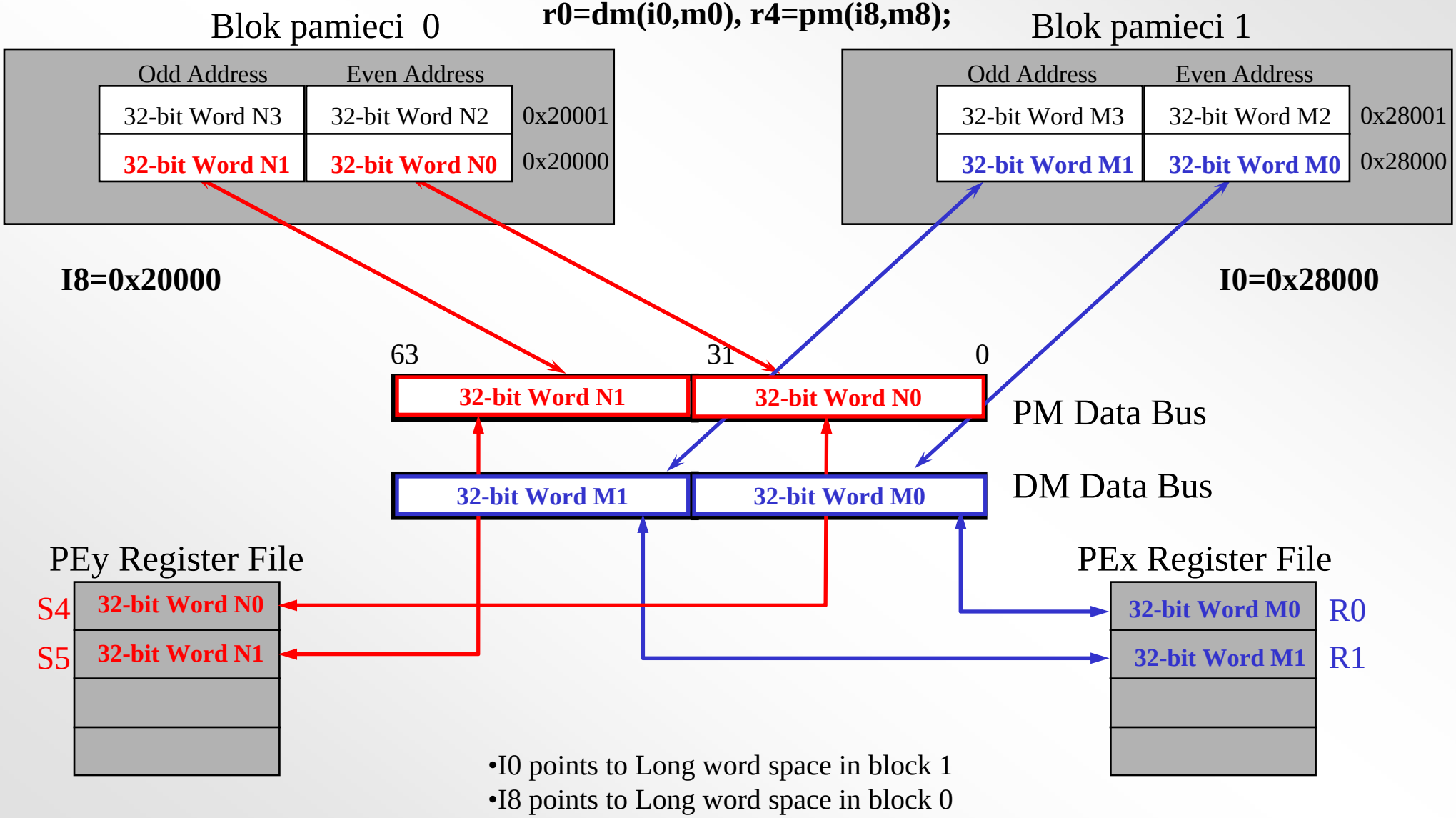
**Przyklad:**
**r0=dm(i0,m0), r4=pm(i8,m8);**




- I0 points to Long word space in block 1
- I8 points to Long word space in block 0

# Long Word Space - Dual Data - SIMD

**Przyklad:**
**r0=dm(i0,m0), r4=pm(i8,m8);**




•I0 points to Long word space in block 1
•I8 points to Long word space in block 0

# ADSP-21161 Dostep do pamieci:zwiekszona precyzja

- **Zwiekszona precyzja VS. Normalna precyzja**
  - –Zwiekszona precyzja ma 40-bity
  - –magistrale ADSP-21161 sa 64-bitowe, dlatego slowo ze zwiekszono precyzja moze zostac przeslane kazda z nich w pojedynczym cyklu podczas gdy 2 slowa ze zwiekszona precyzja bylyby transferem 80-bitowym
  - – maksymalnie 4 slowa danych o normalnej precyzji moga byc przeniesione z/do pamieci w cyklu a slowa ze zwiekszona precyzja (kontrolowane przez IMDWx w SYSCON) tylko 2
  - – Dostep do pamieci normalnych slow ze zwiekszona precyzja za pomoca trybu SISD lub SIMD
  - –ustawienia IMDWx wplywaja tylko na dostep normalnych slow do adresu

# Dostep do danych o róznej dlugosci

Linie obu magistral  nie musza przesylac slow o tej samej dlugosci

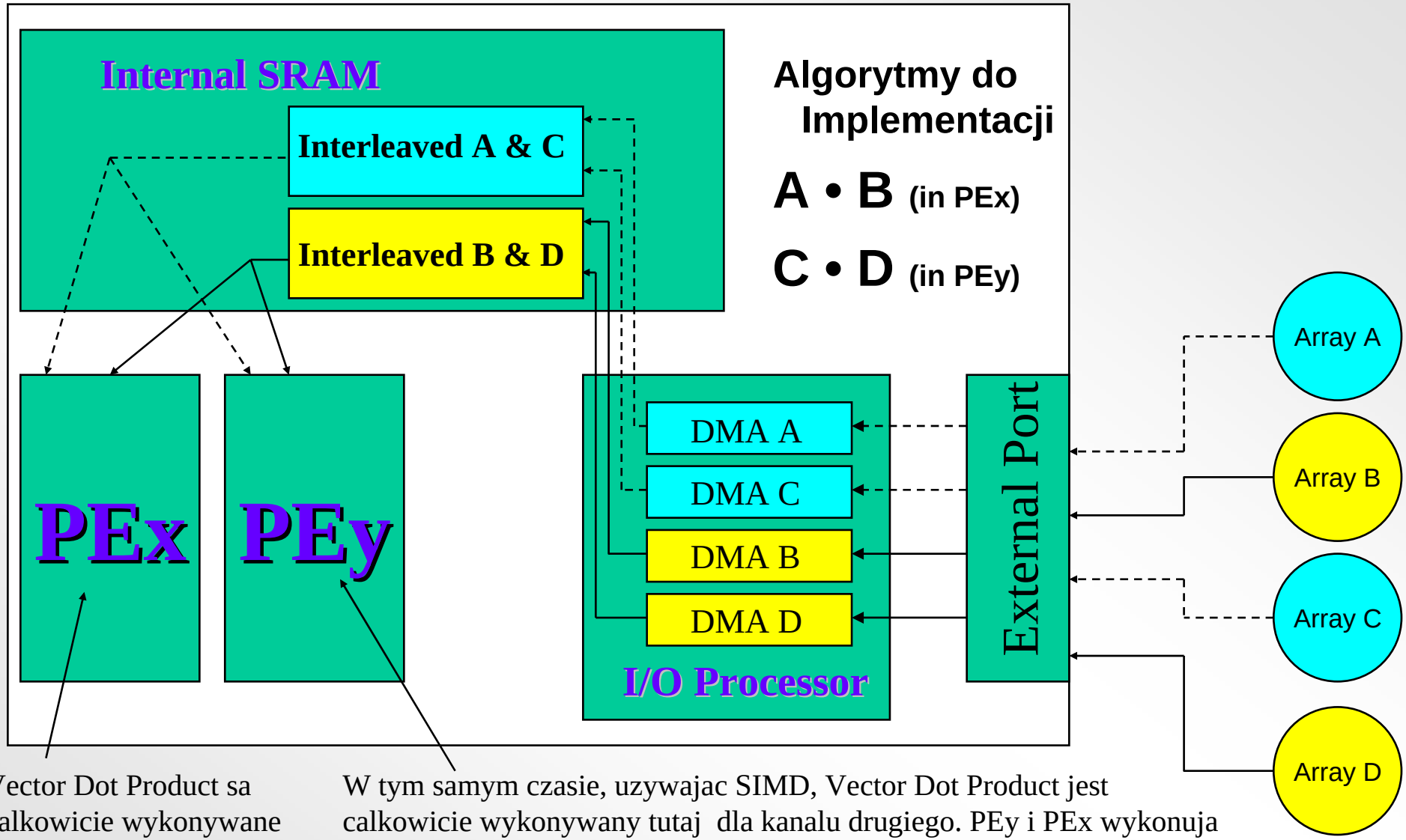
- **Szerokosc slow podwojnych danych  w trybie SISD**
  - architektura pamieci pozwala na mieszanie wszystkich kombinacji dual-data trybu SISD  Short word, Normal word, extended precision Normal word, i dostepu Long word
- **Szerokosc slow podwojnych danych w trybie SIMD**
  - architektura pamieci pozwala mieszac slowa dual data Short trybu SIMD i dostep Normal word  **- L U B -** slowa extended precision Normal i  slowa dostepu Long word
  - ***zadne inne kombinacje  slow podwojnej danej trybu dostepu SIMD sa niedopuszczalne*.**
  - Symulator 21161 oznaczy  niepoprawnie zmieszane slowa dual data trybu dostepu SIMD  jako blad

# ADSP-21161 SIMD Model  programowania

# Dwie drogi programowania z SIMD

- **Multi-Channel Data Optimization**
  - –Cale funkcje lub programy sa wykonywane na kazdym przetwarzajacym elemencie rownolegle
  - –po drugie przetwarzanie jest wykonane w tym samym czasie
  - – PEx operuje na channel 1 a PEy operuje na channel 2
  - –wymaga dostepnosci rownolegle dwóch kanalów danych
  - –Przyklad: dwa kanaly stereo audio -wykonaj filtracje na jednym kanale audio audio w PEx i rownolegle wykonaj filtracje na drugim kanale audio w PEy
- **Single-Channel Data Optimization**
  - –gleboko zagniezdzone bloki instrukcji, i.e.,wewnetrzne petle, sa wykonywane rownolegle
  - –ta sama ilosc przetwarzania jest zrobiona w polowie czasu
  - –nie moze bys zaleznosci danych pomiedzy kolejnymi iteracjami wewnatrz petli , i.e., dane do pracy musza byc niezalezne

# Multi-Channel Data Optimization



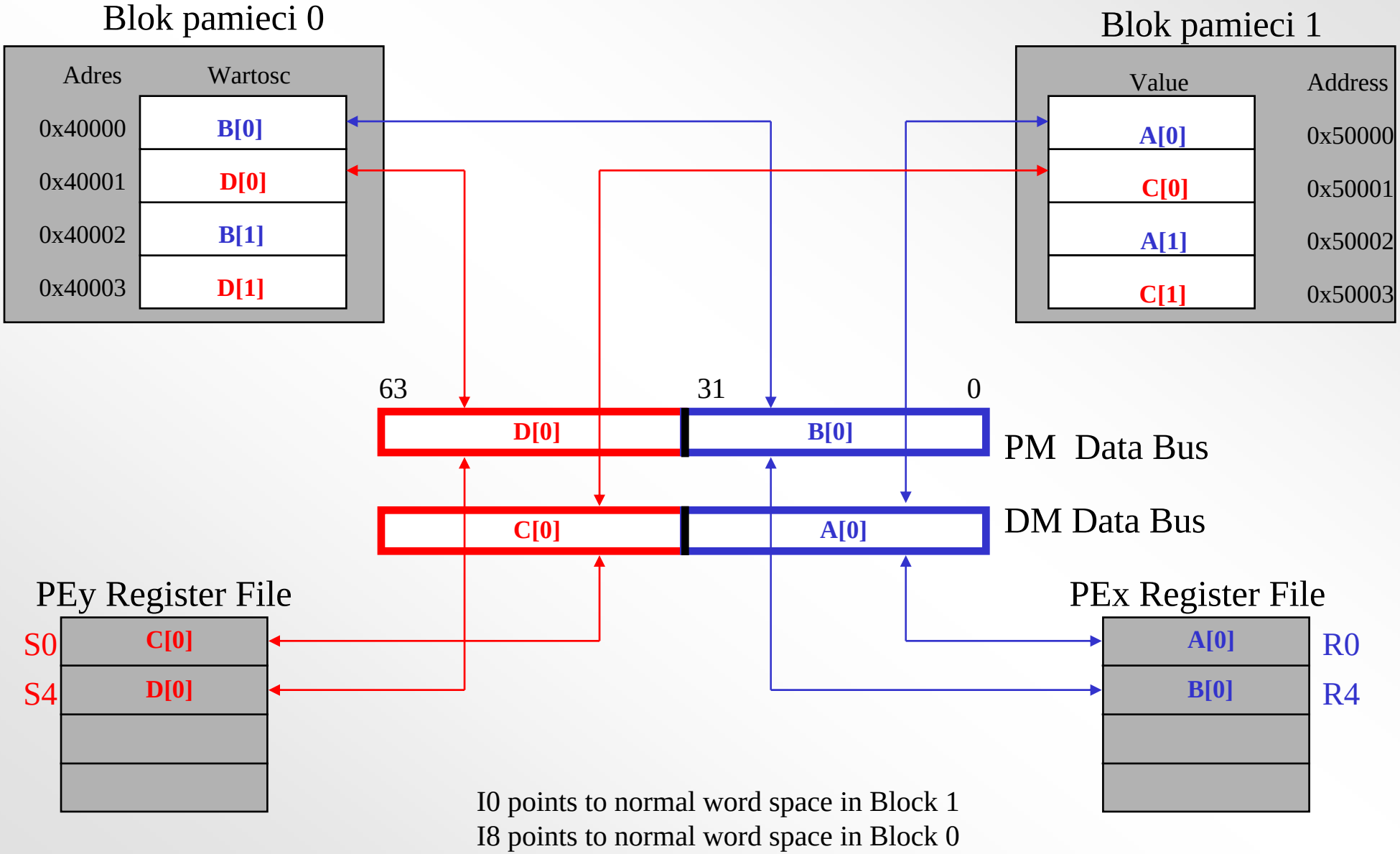

Vector Dot Product sa calkowicie wykonywane tutaj dla kanalu pierwszego

W tym samym czasie, uzywajac SIMD, Vector Dot Product jest calkowicie wykonywany tutaj dla kanalu drugiego. PEy i PEx wykonuja petle Vector Dot Product N-3 razy.

# Przelozenie danych dla Multi-Channel Optimization

Dot Product Code: SIMD Enabled

f8=f0*f4, f12=f8+f12, f0=dm(i0,m1), f4=pm(i8,m9);



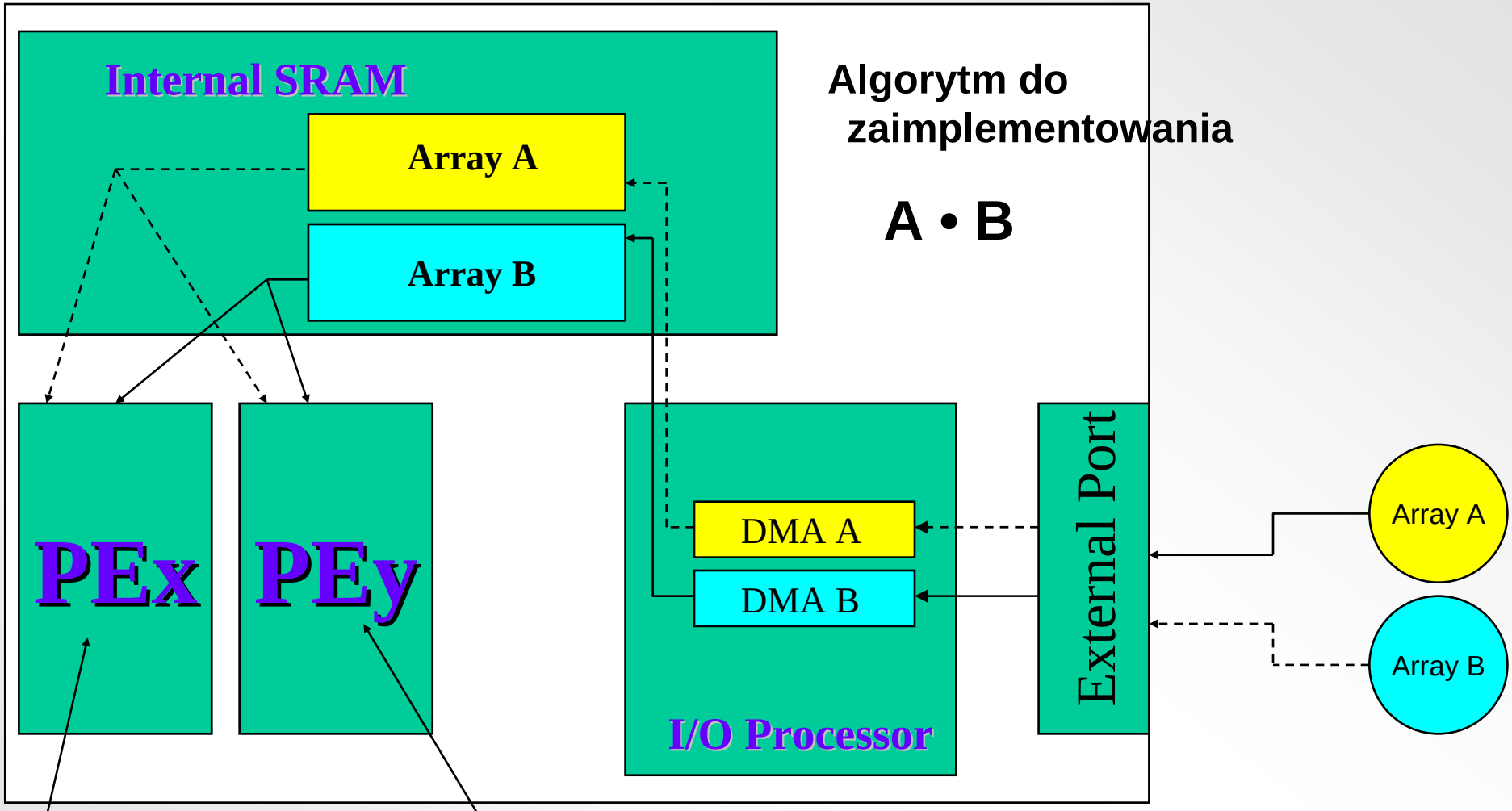

I0 points to normal word space in Block 1

I8 points to normal word space in Block 0

# Optymalizacja danych Multi-Channel dla Vector Dot Product

```
#include “def21161.h”
.section/pm seg_pmco;

/* Inicjalizacja wskaznikow  */
B0 = INPUT_A_C;       /* Przelozone macierze A i C  */
L0 = 0;

B8 = INPUT_B_D;       /* Przelozone macierze B i D  */
L8 = 0;

M1=2;        /* Linia zmodyfikowana, bylo;  “M1=1;”dla SIMD fetch */
M9=2;        /* Linia zmodyfikowana, bylo: “M9=1;”dla SIMD fetch */

/* Uruchomienie trybu SIMD  */
bit set MODE1 PEYEN;              /* Linia dodana zeby uruchomic tryb  SIMD  */
nop;                              /* Linia dodana dla MODE1 effect latency */

f0=dm(i0,m1), f4=pm(i8,m9);       /* Fetching A[0]/C[0] over DM and B[0]/D[0] over PM */
f8=f0*f4, f0=dm(i0,m1), f4=pm(i8,m9);
f12=f0*f4, f0=dm(i0,m1), f4=pm(i8,m9);

lcntr=(N-3), do vecprod until lce;       /* Vector product loop */
vecprod:    f8=f0*f4, f12=f8+f12, f0=dm(i0,m1), f4=pm(i8,m9);

f8=f0*f4, f12=f8+f12;             /* ostatnie mnozenie - accumulate */
f12=f8+f12;                       /* Last accumulate *
```

# Single-Channel Data Optimization



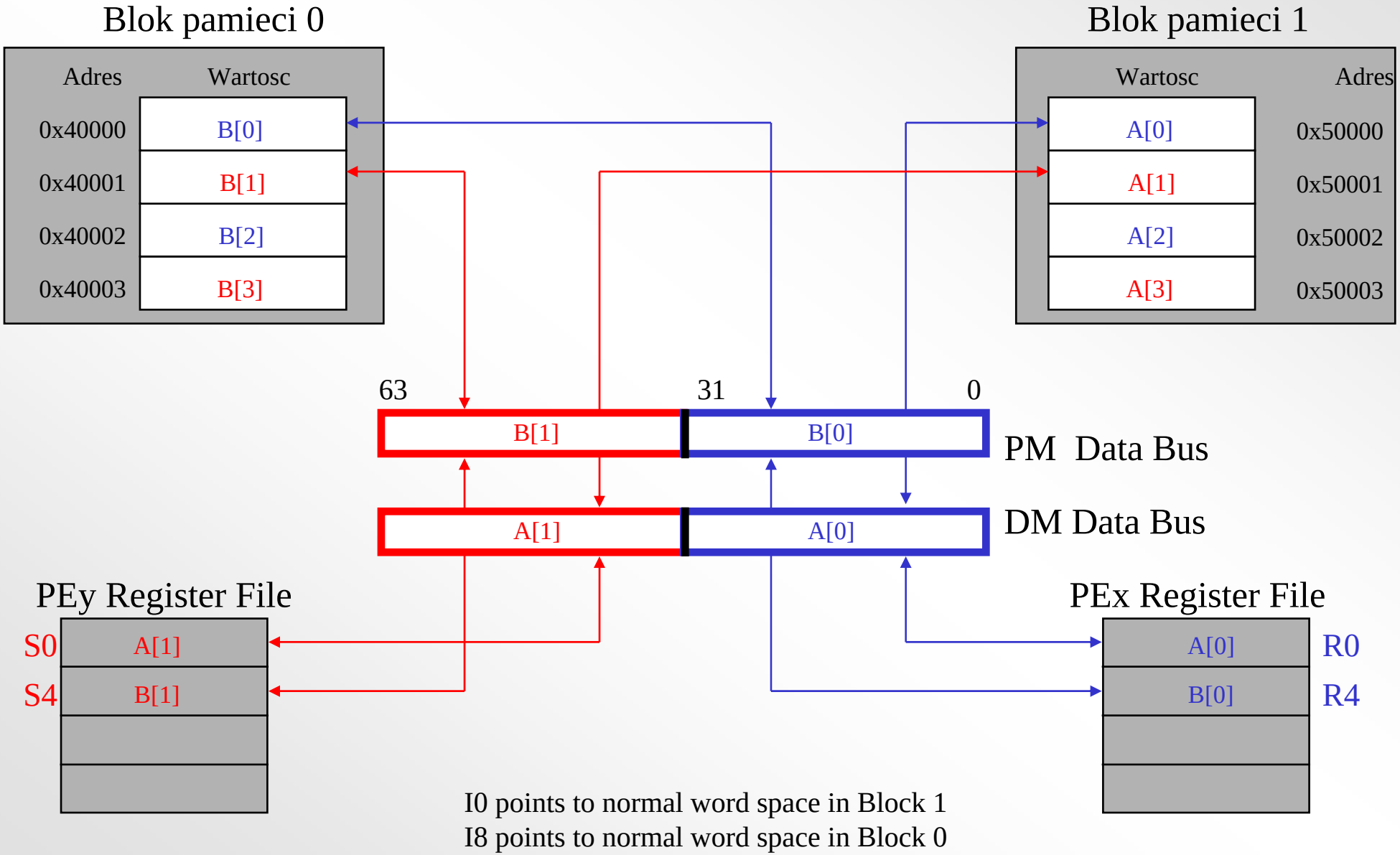

Polowa liczby iteracji , n/2-3, petli w Vector Dot Product jest wykonywana tutaj.

Pozostala polowa iteracji jest wykonywana tutaj. To ma miejsce rownolegle do the iteracji petli w PEx. Uwaga: dane przepuszczone tutaj sa inne niz w PEx. PEy rowniez wykonuje sie n/2-3razy.

# Przelozenie danych dla Single-Channel Optimization

Dot Product Code: SIMD aktywowany( enabled)

f8=f0*f4, f12=f8+f12, f0=dm(i0,m1), f4=pm(i8,m9);




I0 points to normal word space in Block 1

I8 points to normal word space in Block 0

# Optymalizacja danych Single-Channel dla Vector Dot Product

```
#include “def21161.h”
.section/pm seg_pmco;

/* inicjalizacja wskaznikow*/
B0 = INPUT_A;
L0 = 0;

B8 = INPUT_B;
L8 = 0;

M1=2;          /* Linia zmodyfikowana, bylo: “M1=1;”dla SIMD fetch */
M9=2;          /* Linia zmodyfikowana, bylo:“M9=1;”dla SIMD fetch */

/* uruchomienie trybu SIMD  */
bit set MODE1 PEYEN;     /* Linia dodana zeby uruchomic tryb SIMD  */
nop;                                  /* Linia dodana dla MODE1 effect latency */

f0=dm(i0,m1),  f4=pm(i8,m9);          /* Fetching A[0]/A[1] over DM and B[0]/B[1] over PM */
f8=f0*f4,  f0=dm(i0,m1),  f4=pm(i8,m9);
f12=f0*f4,  f0=dm(i0,m1),  f4=pm(i8,m9);

lcntr=(N/2-3), do vecprod until lce;          /* Linia zmodyfikowana, bylo: “lcntr=(N-3),”dla wykonania trybu SIMD   */
vecprod:  f8=f0*f4,  f12=f8+f12,  f0=dm(i0,m1),  f4=pm(i8,m9);

f8=f0*f4, f12=f8+f12;                 /* ostatnie mnozenie – accumulate */
f12=f8+f12;                           /* Last SIMD accumulate */

r8=s12;                               /* Linia dodana do pobrania czesciowej sumy z PEy doPEx */
f12=f8+f12;                           /* Linia dodana zeby polaczyc sumy czesciowe
```

# ADSP-21161 Address Alignment

- **dyrektywa Assemblera .ALIGN**

  **-**Wymusza wyrównanie adresu elementu instrukcji lub danych

  -powinna byc uzywana do przypisania wszystkich macierzy/list dla poprawnego dostepu.

  - Pozwala zeby wszystkie tablice sub-64 bitowe byly dostepne przez przestrzen adresowa long-word , przez przestrzen adresowa normal-word uzywajac opcji LW , lub w trybie SIMD
  - wszystkie macierze tego typu powinny byc przypisane do do 64-bitowego poczatkowego adresu.

  –wywiera wplyw tylko na deklaracje kolejnej zmiennej

  –uzywa wyrazenia: **.ALIGN *wyrazenie*;**

  - Dla slów 32-bitowych, uzyj ".ALIGN 2;"dla slow 16-bitowych, uzyj ".ALIGN 4;".
  - Przyklad: zeby przypisac 32-bitowa macierz nazwana *buff_A* do 64-bitowego poczatkowego adresu, uzyj nastepujacych instrukcji:

```
.SECTION/DM     seg_dmda;
.ALIGN          2;
.VAR            buff_A[N];
```

# Flagi statusu ADSP-21161

## Sa dwa ustawienia flag statusu:

–dla PEx: ASTATX STKYX

–dla PEy: ASTATY STKYY

## Operacje warunkowe w trybie SIMD

– CONDITIONAL COMPUTE(obl. warunkowe): niezalezne obliczenie i wykonanie w PEx i PEy
  – CONDITIONAL DATA MOVES(war. przen. danych): niezalezne obliczenie i wykonanie w PEx i PEy
  – CONDITIONAL BRANCHING: decyzje sa podejmowane przez iloczyn flag statusu
  – DO LOOPS(petle do) : wykonanie jest zalezne od iloczyny flag statusu

# Przyklad #3 programowania

**LAB 13**