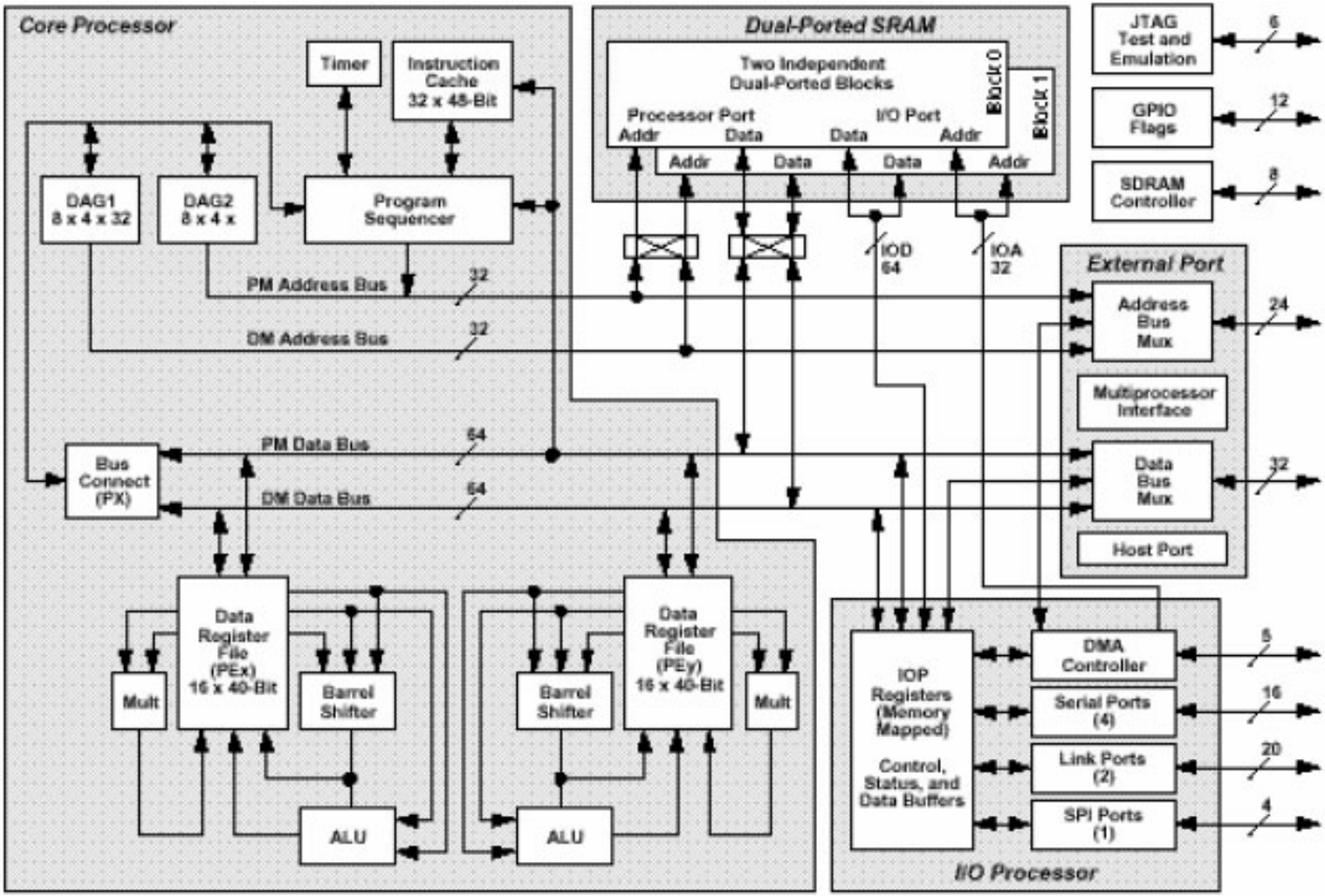
Pamięć ADSP-21161

Sekcja 5

Schemat blokowy ADSP-21161



Pamięć ADSP-21161

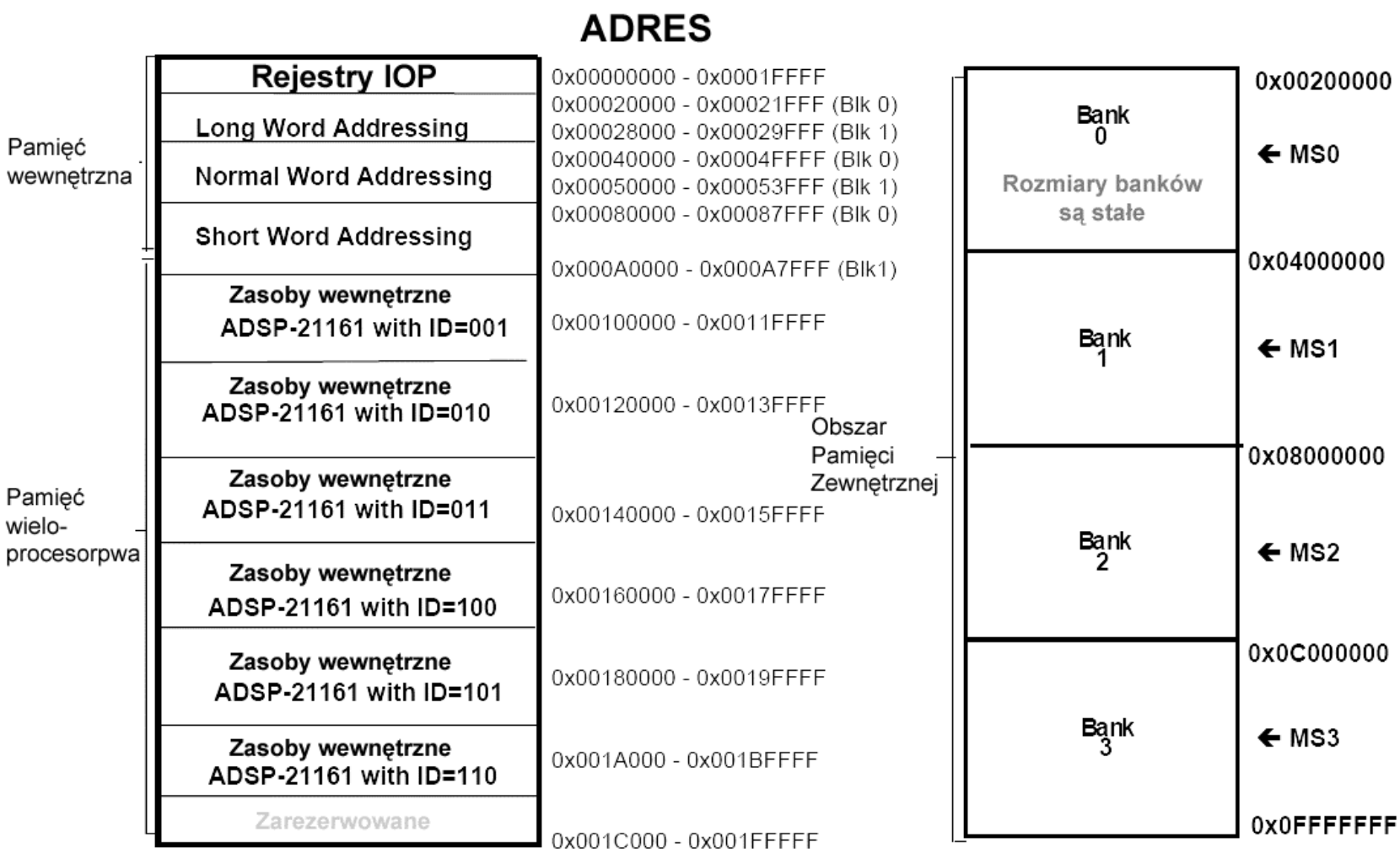
- Przechowuje dane i instrukcje
- 21161s może korzystać z pamięci zewnętrznej, gdy jest dostępna
  - włączając w to peryferia mapowane w pamięci
- 21161s korzysta z pamięci poprzez szyny danych/adresów
  -szyna PMA ma 24 bity szerokości = do 2^24 możliwych lokalizacji instrukcji
- 21161 wykorzystuje liniowe adresowanie
  - 0x0000000 do 0x0FFFFFFF
  - Zakresy pamięci wewnątrz tego obszaru definiują użycie pamięci
    - wewnętrzne bloki pamięci
    - obszar wieloprocesorowy
    - pamięc zewnętrzna

Pamięć ADSP-21161

Struktura Pamięci

- **Dwuportowa pamięc wewnętrzna do niezależnych dostępów dla rdzenia i I/O**
  - **rdzeń i procesor I/O mogą używać pamięci wewnętrznej w tym samym momencie**
- **Pamięć wewnętrzna podzielona na dwa bloki**
  - **obsługuje architekturę z dwoma szynami danych**
  - **do czterech wartości 32-bitowych w jednym cyklu (dwie na szynę)**
- **Wbudowana pamięć SRAM 1MBit**
  - **redukuje ‘wąskie gardła’ I/O**
  - **dwa bloki, po 1/2 Mbit**
  - **konfigurowalna do dostępów 16, 32, 48, 64 bit**

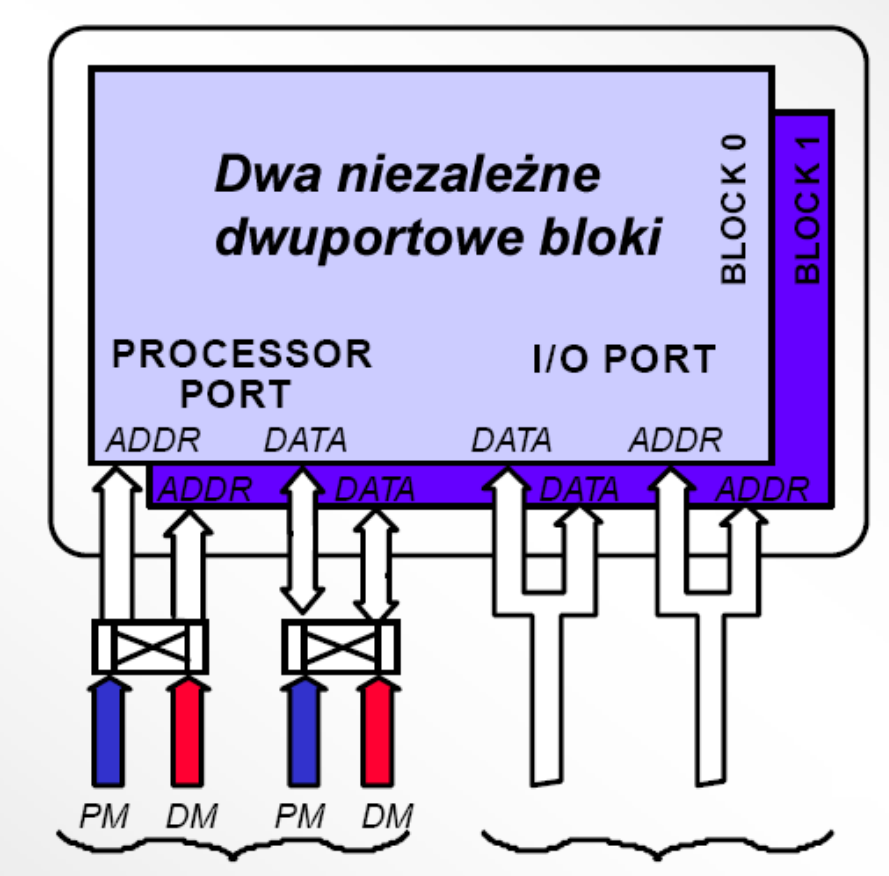
Mapa pamięci ADSP-21161



Organizacja wewnętrzna pamięci

- Pamięć wewnętrzna ADSP-21162 jest konfigurowalna przez użytkownika – pozwala to na bardzo wydajne użycie pamięci do wielu zastosowań
- Każdy blok może być skonfigurowany na przechowywanie tylko instrukcji, tylko danych, lub obydwu jednocześnie, aż do maksymalnego rozmiaru bloku
- Każdy blok jest fizycznie złożony z kolumn, po 16 bitów każda
- Wszystkie dostępy do pamięci są szerokości słowa, szerokość słowa określa jak wiele kolumn i ile bitów jest użytych

Architektura wewnętrzna SRAM



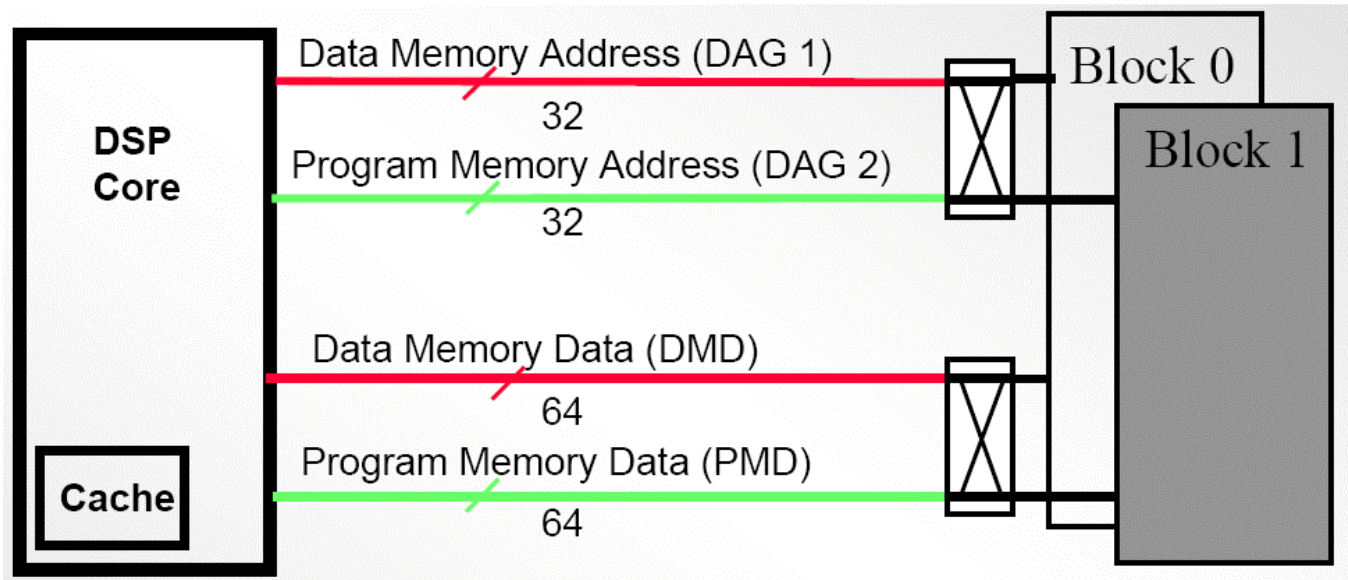
Rdzeń DSP widzi RAM jako dwa bloki

IOP widzi RAM jako jeden ciągły blok

Dwa dostępy rdzenia na cykl

Wydajna architektura SHARC pozwala na zapewnienie transferu nadążającego za obliczeniami

Zmodyfikowana Architektura Harvardzka



Instrukcje są podawane cykl po cyklu po szynie Program Memory, o ile nie znajdują się już w cache. Dostępy do pamięci danych mają pierwszeństwo nad dostępami do pamięci programu. Jeśli pobranie instrukcji, i pobranie danej mają zajść w tym samym cyklu, następuje konfilkt szyny.

Przykłady cyklu dostępu do pamięci

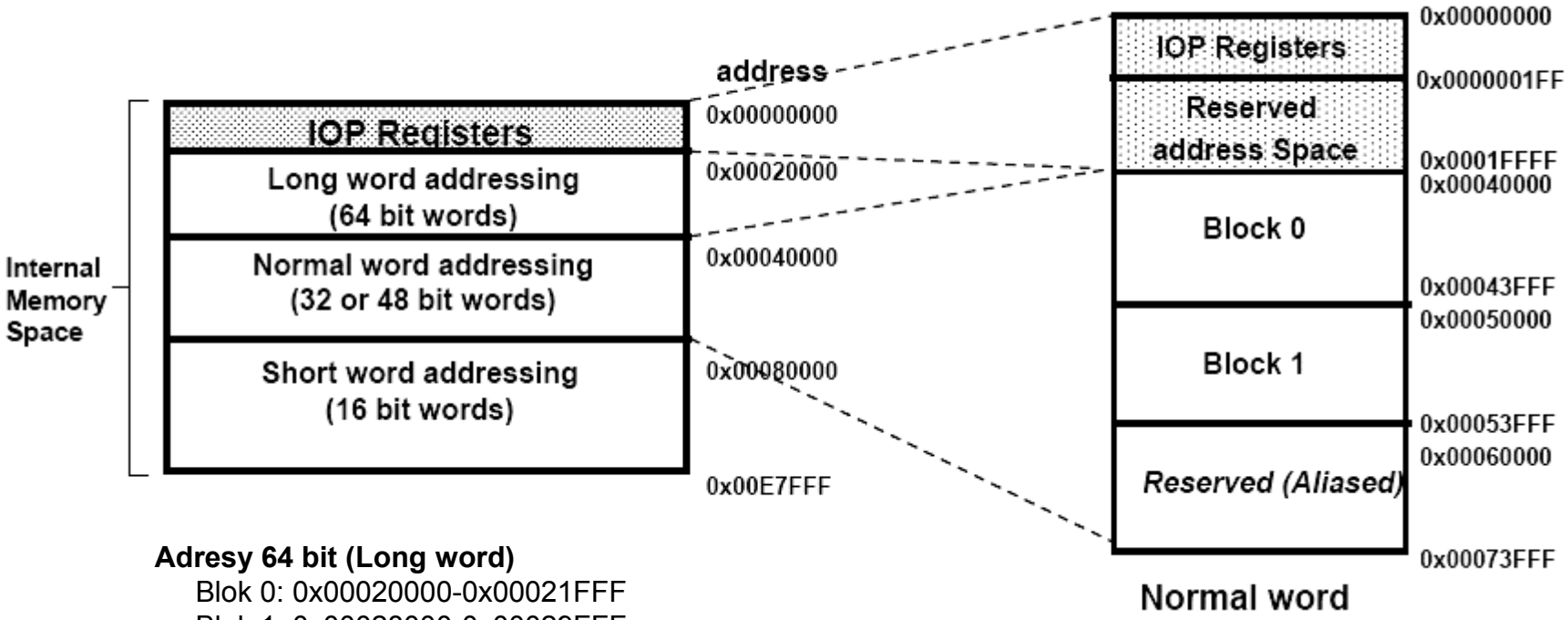
	Instruction	PM Data	DM Data	Core Cycles
1	Block0	None	Block1	1
2	Block0	Block0	Block1	2, 1 if cached
3	Block0	None	Block0	2 always
4	Block0	Block1	Block0	2 always
5	Block0	Block0	Block0	3, 2 if cached
6	Block0	Block1	Block1	3, 2 if cached

Wszystkie przykłady są poprawne

Różnią się czasem wykonania

Zawsze, gdy występuje konfikt pomiędzy dostępem do danych i Dostępem do instrukcji, instrukcja będzie oczekiwać jeden cykl

Konfiguracja wewnętrzna pamięci 21161



Adresy 64 bit (Long word)

Blok 0: 0x00020000-0x00021FFF
Blok 1: 0x00028000-0x00029FFF

Adresy 32/48 bit (Normal word)

Blok 0: 0x00040000-0x00043FFF
Blok 1: 0x00050000-0x00053FFF
Zarezerwowane (alias): 0x000600000-0x00073FFF

Adresy 16 bit (Short word)

Blok 0: 0x00080000-0x00087FFF
Blok 1: 0x000A0000-0x000A7FFF
Zarezerwowane (alias): 0x000C0000-0x000E7FFF

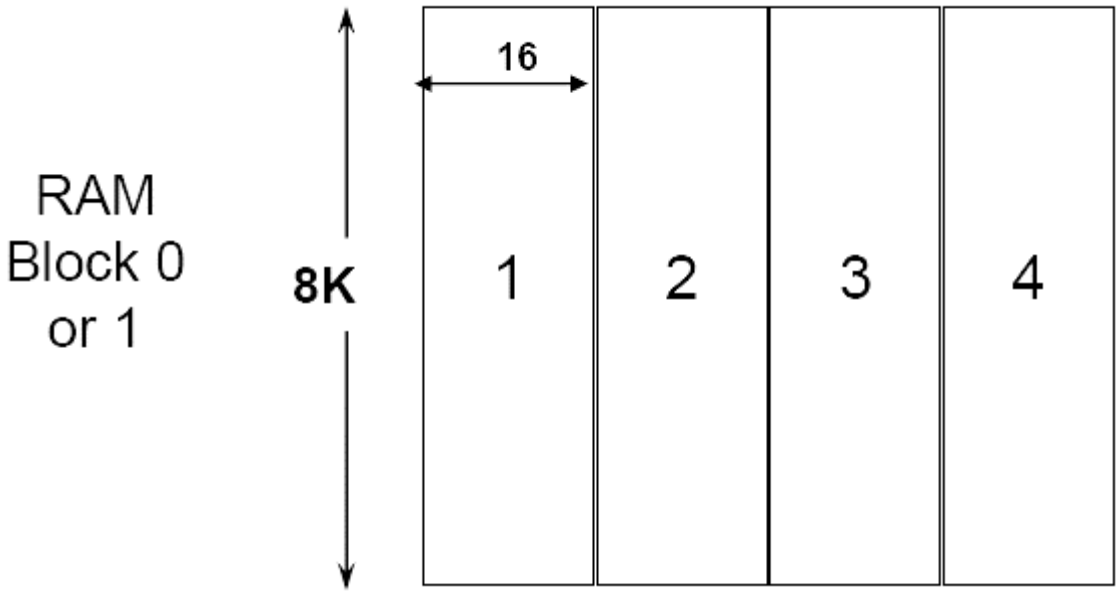
Wewnętrzne adresowanie pamięci

- **Pamięć może być adresowana w blokach 0 i 1, za pomocą trzech trybów adresowania : Long word, Normal word, Short word (długie słowo, normalne słowo, krótkie słowo).**
- **O typie adresowania wnioskuje się z zakresu adresu**
- **Różne metody adresowania opisują tą samą pamięć fizyczną**
- **Przykład:**

  Adres typu Long word: 0x00020000 odpowiada tym samym lokacjom co adres Normal word: 0x00040000 oraz 0x00040001 oraz co adresy Short word: 0x00080000, 0x00080001, 0x00080002 i 0x00080003

Rozmiary bloków wewnętrznych

- **Każdy blok pamięci w ADSP-21161 ma 0.5 Mbit (4 kolumny, każda po 16 bitów, z 8K rzędów)**
  - **-32 bitowe słowo (dane) wymaga dwóch kolumn**
  - **-64 bitowe słowo lub transfery SIMD wymagają 4 kolumn (podczas dostępu do jednego rzędu)**
  - **-48 bitowe słowo (instrukcja) wymaga trzech kolumn**
  - **-40 bitowe słowo (dane) wymaga trzech kolumn.**



Dostęp do pamięci

Są dwa typy odwołań do pamięci, które realizuje rdzeń 21161 (wewnętrzna lub zewnętrzna pamięć)

-Pobranie instrukcji

- **-Kontrolowane z sekwencera programu**
- **-Używające szyny Pamięci Programu (PM)**
- **-Z wewnętrznej (któryś z bloków) lub zewnętrznej pam.**
- **-Instrukcja jest pobiedana co cykl instrukcji (z cache lub pamięci**
- **-zawsze 48 bit szerokości**

-Pobranie danej

- **-Sterowane za pomocą kodu programu**
- **-Przeprowadzane przez DAG**
- **-Używa szyby pamięci programu lub danych (PM lub DM)**
- **-Do/Z wewnętrznej lub zewnętrznej pamięci**
- **-32/64/40 bitów danych (40 bitowe dane wymagają 48bitowego adresowania). Adresowanie danych konfigurowane za pomocą bitów kontrolnych w rejestrze SYSCON**

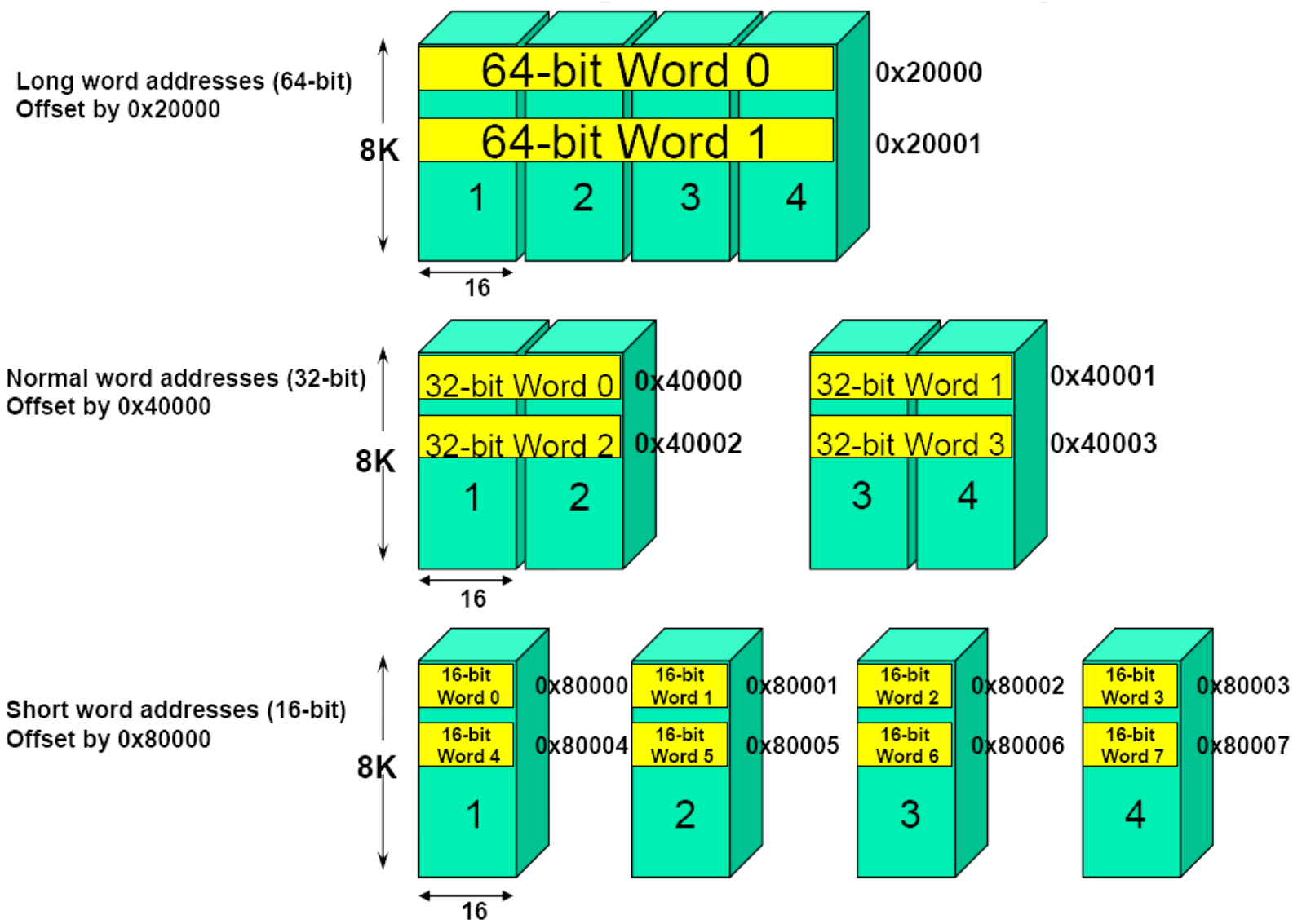
Dostęp do pamięci wewnętrznej

Mapowanie kolumn dla dostępów dla danych, określone jest przez bity lMDW0 i MDW1 (0=32 bit, 1=40bit) w rejestrze SYSCON

- **modyfikuje odwołania typu Normal word**
- **Dane 32 bit wymagają dwóch kolumn**
- **Dane 40 bit wymagają trzech kolumn**

Mapowanie kolumn dla dostępów do instrukcji jest zawsze 48 bitowe, i wymaga trzech kolumn.

Dostęp do pamięci wewnętrznej



Organizacja pamięci a adresy

ADSP-21161 (1Mbit=128 kilobajtów)

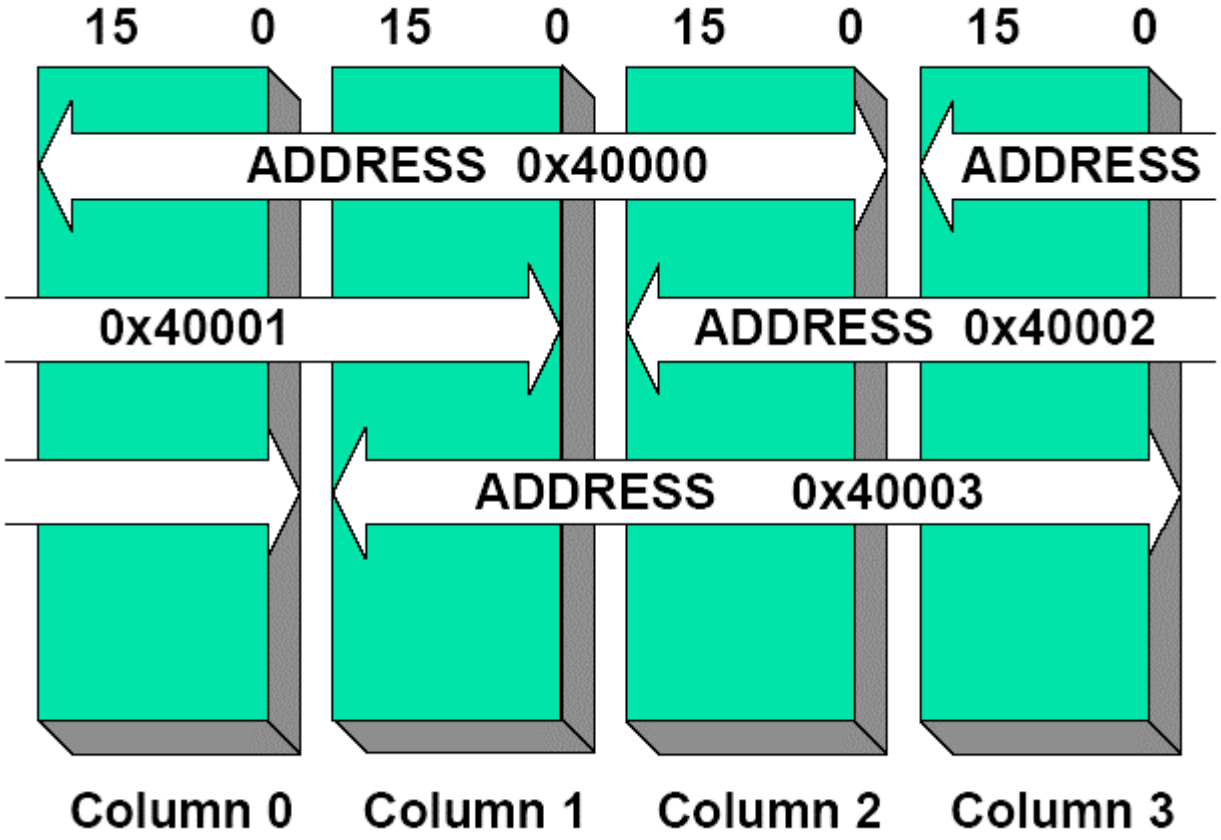
Dwa bloki 0.5 Mbit po 4 kolumny, 16 bit szerokości, 8K rzędów
-Maksymalna liczba 48bitowych słów: 21.325K (10.67K w każdym bloku)
    BLOK 0: (wszystkie słowa 48bitowe) 0x0040000 – 0x0042AA9
    BLOK 1: (wszystkie słowa 48bitowe) 0x0050000 – 0x0052AA9
-Maksymalna liczba 64bitowych słów: 16K (8K w każdym bloku)
    BLOK 0: (wszystkie słowa 64bitowe) 0x0020000 – 0x0021FFF
    BLOK 1: (wszystkie słowa 64bitowe) 0x0028000 – 0x0029FFF
-Maksymalna liczba 40bitowych słów: 21.325K (10.67K w każdym bloku)
    BLOK 0: (wszystkie słowa 40bitowe) 0x0040000 – 0x0042AA9
    BLOK 1: (wszystkie słowa 40bitowe) 0x0050000 – 0x0043AA9
-Maksymalna liczba 32bitowych słów: 32K (16K w każdym bloku)
    BLOK 0: (wszystkie słowa 32bitowe) 0x0020000 – 0x0021FFF
    BLOK 1: (wszystkie słowa 32bitowe) 0x0028000 – 0x0029FFF
-Maksymalna liczba 16 bitowych słów: 64K (32K w każdym bloku)
    BLOK 0: (wszystkie słowa 16bitowe) 0x0080000 – 0x0087FFF
    BLOK 1: (wszystkie słowa 16bitowe) 0x00A0000 – 0x00A7FFF

Aby uprościć mieszanie słów 32 i 48 bitowych w jednym bloku

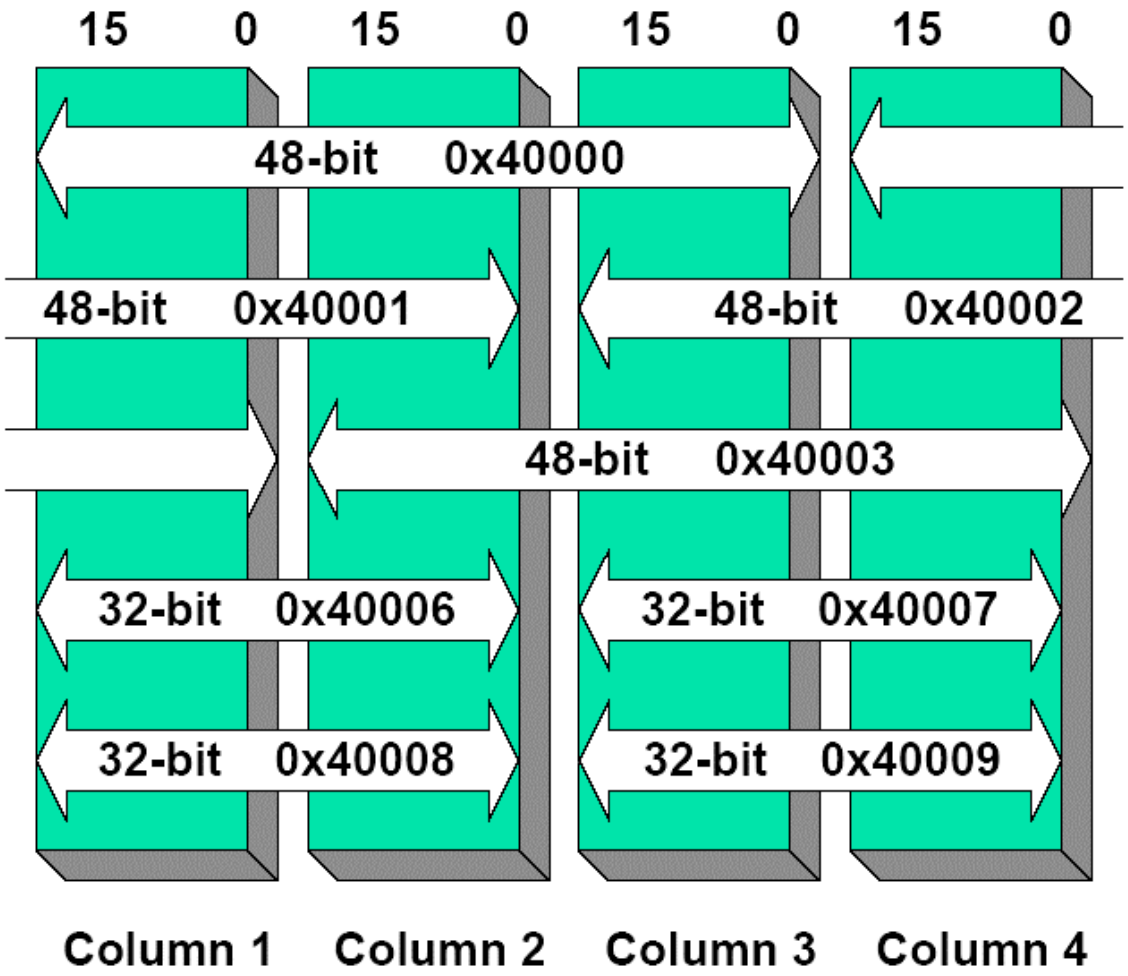
- Mieszanie 32 i 48 bitowych słów realizuje się przez grupowanie kolumn
- Wszystkie instrukcje (48bit) powinny być umieszczone pod niższymi adresami niż dane (32bit) aby zapobiec nakładaniu się słów 48 i 32bit
  - składowanie instrukcji powinno zaczynać się od najniższego adresu w bloku
  - składowanie danych powinno zaczynać się od parzystej kolumny
  - wszystkie dane powinny znajdować się pod wyższymi adresami niż instrukcje
  - instrukcje potrzebują co najmniej trzech ciągłych 16 bitowych kolum
  - słowa danych potrzebują (co najmniej) dwóch 16 bitowych kolumn
- Do opisania grupowania kolumn służy plik linkera

Rotacja słów 48 bit

Rotacja kolumn jest używana aby gęściej upakować słowa 48bit



Mieszanie instrukcji i danych



Przykład: LDF, Segmenty pamięci ADSP-21161

```
MEMORY
{
seg_rth { TYPE(PM RAM) START(0x00040000) END(0x000400ff) WIDTH(48) }
seg_init { TYPE(PM RAM) START(0x00040100) END(0x000401ff) WIDTH(48) }
seg_pmco { TYPE(PM RAM) START(0x00040200) END(0x000419ff) WIDTH(48) }
seg_pmda { TYPE(PM RAM) START(0x00042700) END(0x00043fff) WIDTH(32) }

seg_dm64 { TYPE(DM RAM) START(0x00028000) END(0x000280ff) WIDTH(64) }
seg_dmda { TYPE(DM RAM) START(0x00050200) END(0x00051fff) WIDTH(32) }
seg_heap { TYPE(DM RAM) START(0x00052000) END(0x00052fff) WIDTH(32) }
seg_stak { TYPE(DM RAM) START(0x00053000) END(0x00053fff) WIDTH(32) }
}
```

Mieszanie słów 32bit i 48bit

Wyznaczanie adresów do efektywnego użycia pamięci

m=B+2 [(n MOD 10,922) – TRUNC((n MOD 10,922)/4]

Upraszcza się do

m=B+2[n-TRUNC(n/4)]

Gdzie: n to liczba ciągłych słów 48bit które system przeznaczył na wewnętrzny blok pamięci (n<21845). B to baza adresu (Normal word) wewnętrznego bloku pamieci.
Jeśli {0<n<10,922} to B=0x40000, w przeciwnym wypadku B=0x50000
m to pierwszy 32bit adres (Normal word) do użycia po słowach 48bit

n=0x419FF – 0x40000=0x19FF => 6655 dziesiętnie
B = 0x40000 ponieważ warunek 0 < n < 10922 jest spełniony
m = 0x40000 + 2 [(6655 MOD 10922) - TRUNC((6655 MOD 10922) / 4)]
m = 0x40000 + 2 [6655 - TRUNC(6655 / 4)] = 0x40000 + 9984 dziesiętnie
9984 dziesiętnie = 0x2700 => m = 0x40000 + 0x2700 = 0x4270

Uwaga: poprawne adresy 32Bit danych muszą zaczynać się od parzystych adresów