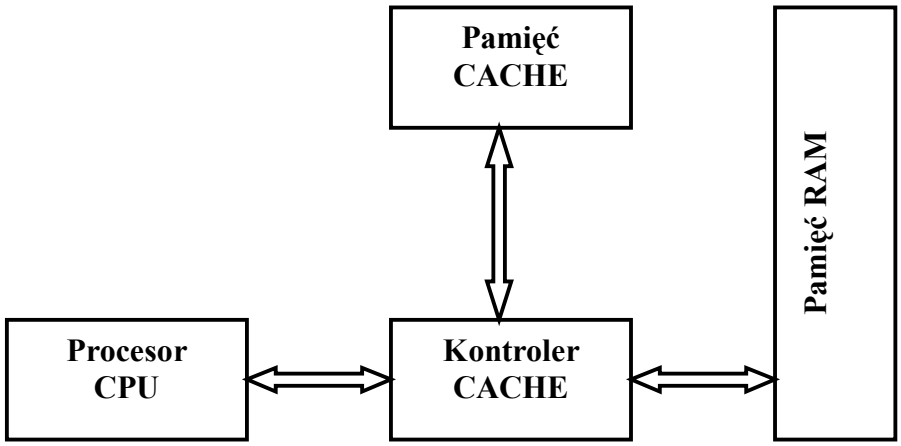
Cel stosowania pamięci cache w procesorach

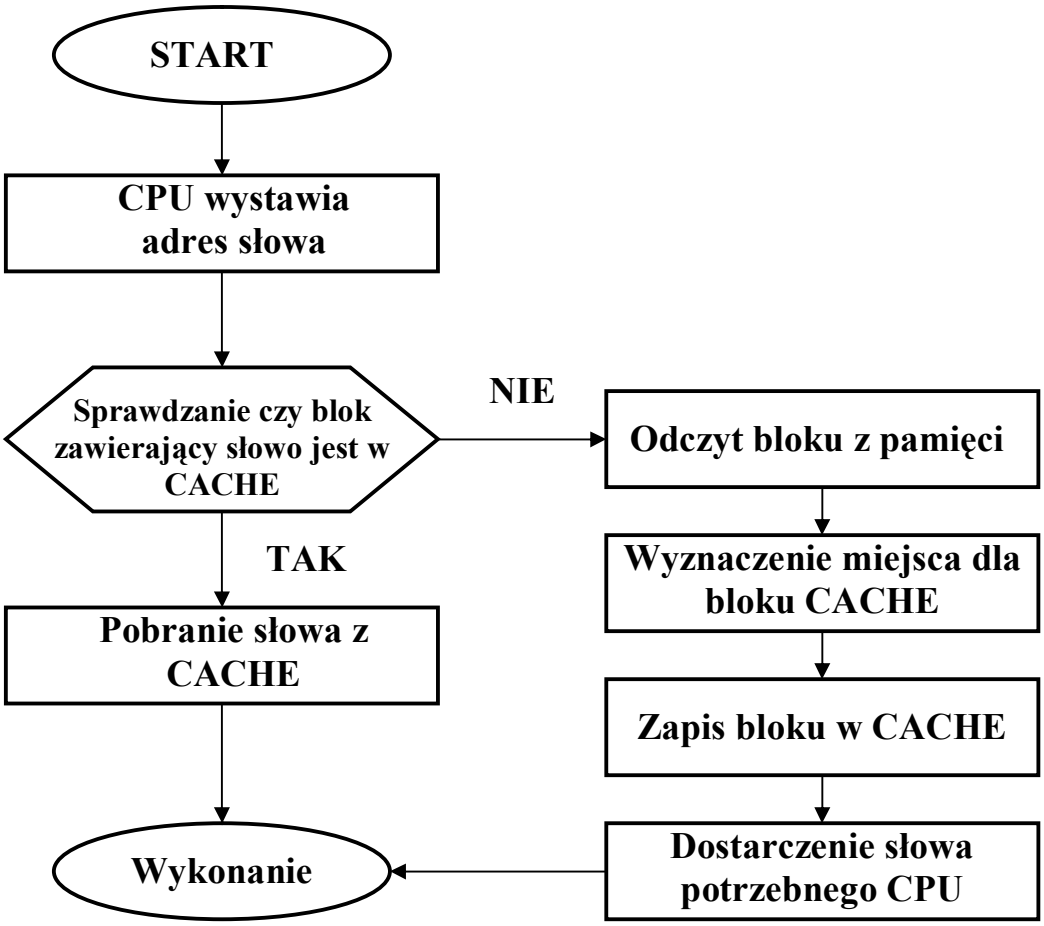
Aby określić cel stosowania pamięci podręcznej cache, należy w skrócie omówić zasadę działania mikroprocesora. Jest on układem cyfrowym taktowanym przez sygnał zegarowy, który realizuje zadany program - ciąg rozkazów umieszczony w pamięci operacyjnej. Program ma zwykle zadanie przetworzenia określonych danych pobranych z pamięci (lub urządzeń zewnętrznych), oraz zapisanie wyników ich przetwarzania, też w pamięci (lub przekazanie do urządzeniach zewnętrznych). Szybkość wykonywania programu zależy w znacznej mierze od czasu dostępu procesora do układu pamięci operacyjnej. Nie bez znaczenia jest także pojemność pamięci (ile danych można w niej zapisać). Stosowane we współczesnych komputerach wielozadaniowe systemy operacyjne umożliwiają uruchamianie wielu programów jednocześnie. Dobrze jest więc gdy procesor ma do dyspozycji dużą pamięć operacyjną RAM. Ważnym czynnikiem pozostaje też koszt zastosowanego układu pamięci. Naturalnie w przypadku praktycznego systemu musi on być jak najniższy.
Istnieją wzajemne zależności pomiędzy wszystkimi opisywanymi wyżej parametrami.

- mniejszy czas dostępu - większy koszt
- większa pojemność - większy czas dostępu

Z tego wynika, że nie jest możliwe wyprodukowanie idealnej pamięci o maksymalnie dużej pojemności, a przy tym małym czasie dostępu i minimalnym koszcie. Możliwe jest budowanie szybkich układów, ale stosunkowo drogich i o małej pojemności. Istnieją też duże pamięci o małych kosztach w przeliczeniu na bajt, ale cechujące się mniejszą efektywnością w zakresie czasu dostępu.
We współczesnych procesorach stosuje się rozwiązanie kompromisowe, polegające na zastosowaniu pamięci wewnętrznej dwupoziomowej. Mikroprocesor wyposaża się we względnie dużą i wolniejszą pamięć główną, oraz w mniejszą, ale szybszą pamięć podręczną cache. Ilustruje to schemat blokowy przedstawiony na rysunku 2. Takie rozwiązanie pozwala na korzystanie z pamięci o dużej pojemności, jednocześnie możliwe jest umieszczenie najpotrzebniejszych danych, w szybkiej pamięci podręcznej.
Sprzętowa pamięć podręczna (ang. Cache memory) jest pamięcią typu SRAM (Static Random Access Memory). Układy tej pamięci są zbudowane z tranzystorów, które trwale przechowują zapisane dane. Pamięć podręczna zawiera kopię części zawartości pamięci głównej. Gdy procesor zamierza odczytać słowo z pamięci, najpierw następuje sprawdzenie, czy słowo to nie znajduje się w pamięci podręcznej. Jeśli tak, to słowo to jest szybko dostarczane do procesora. Jeśli nie, to blok pamięci głównej RAM zawierający określoną liczbę kolejnych słów jest wczytywany do pamięci podręcznej, a następnie potrzebne słowo (zawarte w tym bloku) jest dostarczane do procesora. Następne odwołania do tego samego słowa i sąsiednich zawartych w przepisanym bloku będą realizowane już znacznie szybciej.
Organizacja współpracy procesora z takimi pamięciami wymaga zastosowania dodatkowego układu - kontrolera cache, który steruje tym procesem.



Rys. 1. Podłączenie cache do procesora.



Rys. 2. Algorytm dostępu procesora do pamięci.

Efektywność stosowania cache zależy w znacznej mierze od sposobu ułożenia kodu programów i danych pobieranych z pamięci przez mikroprocesory. Zwykle kod i dane nie są "porozrzucane" przypadkowo po całej dostępnej przestrzeni adresowej w pamięci RAM. Większość odwołań do pamięci w trakcie wykonywania programu odbywa się przez pewien czas pracy mikroprocesora w wąskim obszarze. Zjawisko to jest określane mianem **lokalności odniesień**.

Lokalność odniesień można uzasadnić intuicyjnie w następujący sposób:

- Z wyjątkiem rozkazów skoku i wywołania procedury, realizacja programów ma charakter sekwencyjny. Tak więc, w większości przypadków następny rozkaz przewidziany do pobrania z pamięci następuje bezpośrednio po ostatnio pobranym rozkazie.
- Rzadkością jest występowanie w programach długich, nieprzerwanych sekwencji wywołań procedury (procedura wywołuje procedurę itd.), a potem długiej sekwencji powrotów z procedur. Wymagałoby to ciągłego odwoływania się do oddalonych od siebie obszarów pamięci głównej, zawierających kody poszczególnych procedur.
- Większość pętli (konstrukcji bardzo często występujących w programach) składa się z małej liczby wielokrotnie powtarzanych rozkazów. Podczas iteracji następuje kolejne powtarzanie zwartej części programu.
- W wielu programach znaczna część obliczeń obejmuje przetwarzanie struktur danych, takich jak tablice lub szeregi rekordów ułożone kolejno w pamięci operacyjnej. Tak, więc procesor pobiera dane zapisane w sposób uporządkowany w małym jej fragmencie.

Oczywiście w długim czasie wykonywania programu procesor potrzebuje dane rozmieszczone w różnych odległych miejscach pamięci. Zwykle jednak, po wykonaniu skoku następne odniesienia odbywają się już lokalnie. Przepisanie bloku kolejnych komórek pamięci do szybkiego układu cache może, więc skutecznie przyspieszyć dostęp do pamięci.

W stosowanych obecnie rozwiązaniach można wyróżnić następujące poziomy pamięci podręcznej.

- L1 - (level 1) zintegrowana z procesorem - umieszczona wewnątrz jego struktury.
- L2 - (level 2) umieszczona w jednej obudowie układu scalonego mikroprocesora lub na wspólnej płytce hybrydowej (Pentium II).
- L3 - (level 3) występuje w bezpośrednim sąsiedztwie procesora ITANIUM.

Pamięć podręczna najniższego poziomu (L1 – Level 1) jest stosunkowo mała, ale dane w niej zgromadzone są szybko dostępne dla procesora. W wypadku braku potrzebnych w danym momencie danych (braku trafienia), następuje odwołanie do pamięci kolejnych, wyższych poziomów. Po ich odczycie następuje przepisanie do niższych poziomów, tak by były szybciej dostępne w kolejnych odwołaniach. Jeśli dane nie są aktualnie buforowane w cache, następuje odczyt bloku pamięci głównej RAM, który je zawiera i wymiana zawartości cache. Pamięci niższych poziomów mogą mieć mniejszą pojemność i być bardziej efektywne. Procesor może szybko odczytywać mniejsze porcje danych w jednym cyklu zegara. Wyższe poziomy cache mają większe pojemności, dzięki czemu odwołania do RAM mogą odbywać się rzadziej. Można też w jednym odczycie przepisać większą porcję danych z RAM.

Ważnym zagadnieniem, jest **podział pamięci podręcznej na oddzielny blok dla kodu programu** i oddzielny blok dla danych. W taki sposób jest podzielona pamięć poziomu L1 procesora ITANIUM - Pamięć cache poziomów L2 i L3 jest już wspólna dla rozkazów i danych. W wielu (szczególnie starszych) procesorach wykorzystuje się również jednolitą pamięć podręczną poziomu L1. Takie rozwiązanie też posiada pewne zalety. Poniżej przedstawiono korzyści płynące z zastosowania pamięci oddzielnej i jednolitej.

Pamięć oddzielna kod i dane

Eliminowana jest rywalizacja o dostęp do pamięci między układem pobierania i dekodowania rozkazów w procesorze, a jednostką wykonującą w tym samym czasie inne, poprzednio pobrane rozkazy, które mogą wymagać odczytu pewnych zmiennych z cache. Ma to szczególne znaczenie w przypadku procesorów superskalarnych, w których kilka rozkazów jest wykonywanych równolegle. Dlatego we współczesnych procesorach poziom pamięci podręcznej L1 jest zawsze dzielony na blok danych i instrukcji.

Pamięć łączna dla kodu i danych

Poprawia się współczynnik trafień w tak zorganizowanej pamięci podręcznej, dzięki temu, że naturalnie równoważy się zapotrzebowanie na przechowywanie rozkazów i danych. Jeśli na przykład program wymaga ciągłego pobierania rozkazów i w małym stopniu korzysta z danych, dostępna pamięć podręczna zapełni się w większości rozkazami. Oddzielna cache dla danych w takiej sytuacji pozostałaby niewykorzystana. Drugą zaletą jest to, że upraszcza się układ procesora - łatwiej jest zrealizować w jego strukturze jeden bufor pamięci cache niż dwa.

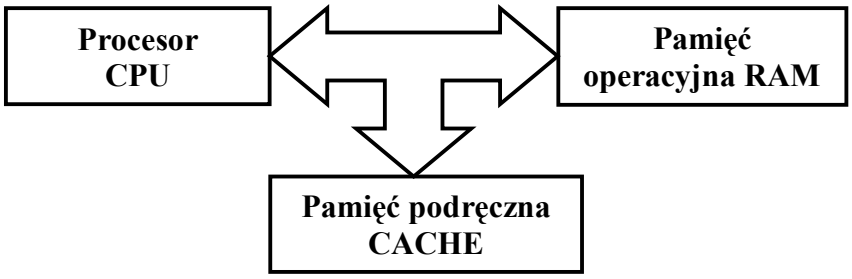
Sposoby dołączania pamięci cache do procesora

Układ cache jest w obecnych procesorach ściśle związany z ich strukturą - właściwie poziom L1 fizycznie stanowi integralną część mikroprocesora. Jednak aby łatwiej było przedstawić zasadę działania i sposoby dostępu procesora do pamięci podręcznej, ta część programu traktuje ją jako oddzielny blok logiczny, dołączony do mikroprocesora, nie zajmując się jej fizycznym

umiejscowieniem. Pisząc o sposobach dołączania, mamy więc na myśli sposób umieszczenia bloku cache na drodze procesor - pamięć.

Obecnie stosuje się trzy podstawowe sposoby dostępu procesora do pamięci podręcznej:

Look – Aside (dostęp bezpośredni)

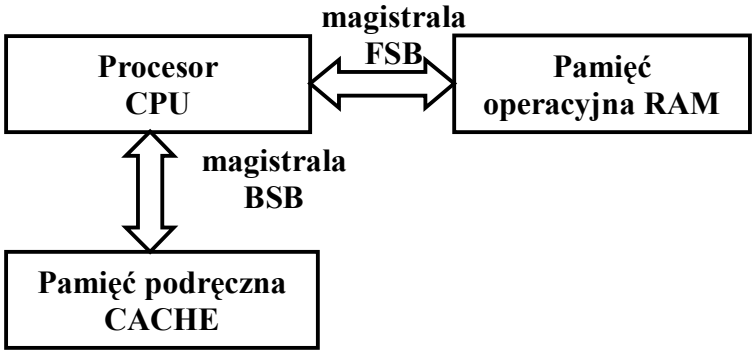


Procesor odwołuje się do cache wykorzystując magistralę pamięciową. Pamięć podręczna jest podłączona równolegle z pamięcią operacyjną RAM. W takim układzie częstotliwość pracy obu pamięci jest taka sama (komunikacja odbywa się po wspólnej magistrali), tylko czas dostępu dzięki szybkości cache może ulec skróceniu. Wykorzystanie tej samej magistrali nie jest korzystne. Jest ona blokowana przy każdym dostępie procesora do cache i nie może być w tym samym czasie udostępniona innym urządzeniom

Look – Throgh (dostęp „przez")

Układ pamięci podręcznej pośredniczy w dostępie procesora do RAM. Procesor odwołuje się do układu cache, natomiast ten układ jest dołączony przez magistralę pamięciową do RAM.

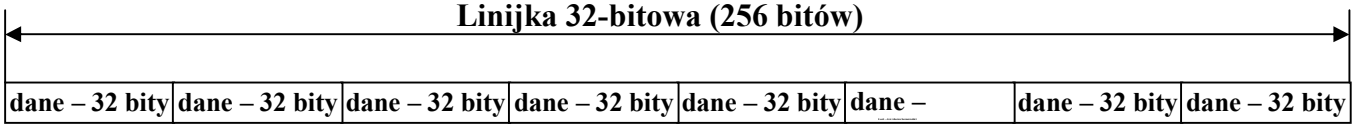
Look – Backside (dostęp od tyłu)



Ulkład pamięci podręcznej jest dołączony do procesora przez oddzielną magistralę nazywaną BSB (*Back Side Bus*). Druga magistrala FSB (*Front Side Bus*) łączy procesor z pamięcią główną. W tym układzie częstotliwości obu magistral są zupełnie niezależne. Możliwe jest też wykorzystanie magistrali FSB przez inne urządzenia zapisujące do pamięci RAM, w czasie gdy procesor komunikuje się z cache po BSB.

Budowa i organizacja pamięci podręcznej

Pamięć cache jest zorganizowana w linijki (o rozmiarach 16 lub 32 bajty - 128 lub 256 bitów), w których są przechowywane informacje pobrane z RAM w postaci słów binarnych. Jedna linijka jest najmniejszą porcją informacji - blokiem danych jaki układ cache wymienia z pamięcią operacyjną RAM. W różnych linijkach może być więc zapisana kopia zawartości odległych bloków z pamięci głównej.
Poniżej opisano budowę 32-bajtowej linijki pamięci podręcznej L1 w procesorze Pentium (rys. 3). W innych procesorach mogą być zastosowane trochę inne rozwiązania, jednak ogólna zasada organizacji cache w linijki pozostaje taka sama.



Rys. 3. Organizacja linijki w procesorze Pentium.

Procesor Pentium posiada oddzielną pamięć podręczną poziomu L1 dla kodu programu (8kB) i oddzielną dla danych (8kB). Każda z nich jest podzielona na 256 linijek (256 x 32 B = 8kB). Aby zbiór takich linijek był dla procesora użyteczną strukturą, w której łatwo odszukać potrzebne dane, musi istnieć mechanizm zapisywania i kodowania dodatkowych informacji na temat każdej linijki. Przede wszystkim potrzebna jest informacja o tym, które fragmenty zawartości pamięci RAM są aktualnie skopiowane w poszczególnych linijkach. Jest to niezbędne, aby podczas żądania procesora odczytu z pamięci, kontroler cache mógł poprawnie określić czy dane są dostępne w linijkach, czy trzeba je sprowadzić z RAM. Wszystkie te informacje przechowuje się w katalogu cache (czasem określanym skrótem TAG-RAM). Jest on częścią pamięci podręcznej, która zawiera rekordy odpowiadające każdej linijce cache. Są w nich zakodowane informacje na temat danych aktualnie zapisanych w odpowiednich linijkach. Wartości poszczególnych pół tych rekordów mogą także wskazywać, że dana linijka jest wolna.
Podczas dostępu procesora do cache układy logiczne dzielą przekazywany przez niego adres na następujące części:
- Znacznik (20 bitów) jest porównywany ze znacznikiem w katalogu cache. Na podstawie porównania znaczników określa się, czy potrzebne procesorowi dane są w linijce cache (określanie trafienia).
- Wiersz (7 bitów) określa, która pozycja (indeks) w katalogu 1 i katalogu 2 może odwzorowywać potrzebne dane.
- Słowo (3 bity) pozwala na określenie, które z ośmiu 32-bitowych słów przechowywanych w linijce zawiera dane potrzebne procesorowi.
- Bajt (2 bity) określa, który bajt w 32- bitowym słowie jest aktualnie potrzebny mikroprocesorowi.

W wypadku braku trafienia, tak zbudowany adres wskazuje blok w pamięci operacyjnej RAM, zawierający potrzebne dane. Zgodnie z zasadą działania cache, po odczytaniu zawartości bloku z RAM musi ona być zapisana do pamięci podręcznej. Na podstawie bitu LRU oraz części adresu (pola wiersz) jest wyznaczana linijka, do której można dokonać zapisu.

Skrót MESI używany do określania bitów w katalogu cache, został utworzony od pierwszych liter angielskich określeń czterech możliwych stanów linijki (Modified, Exclusive, Schared, Invalid). Na podstawie stanu bitów MESI można także określić, czy dane w poszczególnych linijkach są także zapisane w innych poziomach pamięci podręcznej. Stan bitów MESI zmienia się również podczas modyfikacji danych w pamięci. Dotychczas omawiane były jedynie zagadnienia związane ze skróceniem czas dostępu do danych i rozkazów podczas ich odczytu. Jednak procesor nie tylko odczytuje z pamięci RAM. W trakcie wykonywania programu musi też tam zapisywać wyniki swoich peracji, modyfikować pewne zmienne i dane. Niektóre z nich mogą być w tym czasie skopiowane także do pamięci podręcznej. Wiąże się z tym konieczność zadbania o aktualność obu kopii danych.

Do pamięci operacyjnej oprócz procesora mogą też mieć dostęp inne urządzenia. Mogą one zapisywać i odczytywać RAM bez udziału mikroprocesora. Jeśli dokonają zmiany słowa w RAM, którego kopia aktualnie jest przechowywana w cache, mogą spowodować, że procesor posiada w pamięci podręcznej nieaktualne dane. Również gdy procesor w trakcie programu dokona zmiany danych tylko w cache, inne urządzenia mogą odczytać bezpośrednio z RAM stare błędne wartości. Należy zaznaczyć, że taka niespójność może występować tylko dla pamięci podręcznej danych. W oddzielnej pamięci podręcznej kodu programu procesor nie zapisuje swoich wyników, gdyż przechowuje ona jedynie dla niego instrukcje - kolejne rozkazy. Są różne rozwiązania problemu spójności danych stosowane w różnych procesorach, jednakże generalnie można wyróżnić dwa sposoby:

Write Trougch (zapis jednoczesny)

W tym sposobie każdy zapis danych wykonywany jest jednocześnie zarówno do pamięci głównej jak i do cache. Każdy zapis wymaga więc dostępu procesora do pamięci RAM. Również każdy bezpośredni zapis do RAM wykonywany przez inne urządzenia musi być monitorowany przez procesor. W ten sposób może on w razie potrzeby uaktualnić zawartość cache. Jest to więc najbardziej naturalny sposób, jednak generuje znaczny przepływ danych między pamięciami co powoduje duże opóźnienia.

Wirte Back (zapis opóźniony)

W tym trybie przy zapisie procesor aktualizuje tylko pamięć podręczną. Jednocześnie dla zmodyfikowanych linijek układ cache ustawia odpowiednie statusy (na bitach MES). Zawartość pamięci głównej jest aktualizowana później na żądanie. Może ono być wyrażone przez instrukcję programową WBINVO (Write Back and Invalid Data Cache), lub specjalny sterujący sygnał sprzętowy. Aktualizacja jest też wyzwalana w wyniku braku trafienia w fazie odczytu z pamięci głównej. Gdy trzeba dokonać wymiany linijki cache, zawartość linijki usuwanej mającej status zmodyfikowany (zakodowany na bitach MESI), musi być koniecznie zapisana do RAM.

Taka implementacja sposobu utrzymania spójności danych jest bardziej wydajna, minimalizuje ilość cyklów zapisu do pamięci głównej. Problemem jest jednak to, że bezpośredni dostęp zewnętrznych modułów wejścia-wyjścia do RAM także powoduje konieczność uaktualniania pamięci cache co może powodować pewne opóźnienia.

W trakcie wykonywania programu może też nastąpić konieczność zapisu danych w obszarach RAM, które nie są aktualnie skopiowane do pamięci podręcznej. Niektóre procesory w takim wypadku mogą po prostu dokonywać zapisu w RAM z pominięciem układu cache. W nowszych generacjach procesorów stosuje się mechanizm, w którym zapis danych pociąga za sobą skopiowanie odpowiedniego bloku RAM do linijki cache, gdzie jest on modyfikowany.

Dzięki temu ewentualny odczyt lub zapis tych samych danych w następnych rozkazach procesora może przebiegać już bez konieczności odwoływania się do RAM.

Praktyczne rozwiązania

Zestawienie i krótki opis rozwiązań w zakresie układu pamięci podręcznej cache spotykanych w popularnych procesorach.

AMD K6 III

- oddzielna pamięć podręczna L1 dla kodu programu i danych
- pamięć podręczna L2 256kB Write Back, 4 - kanałowa
- obydwie L1 są 2-kanałowe
- rozmiar pamięci podręcznej L1 kodu 32kB i danych 32kB
- pamięć L1 danych - zapis write back

AMD K7

- oddzielna pamięć podręczna L1 dla kodu programu i danych po 64 kB
- obydwie 2 kanałowe - odwzorowanie sekcyjno-skojarzeniowe
- rozmiar pamięci podręcznej L2 - 512 kB
- dostęp do L2 oddzielną magistralą BSB

Intel Pentium III

- po 16 kB pamięci L1 oddzielnie dla kodu i danych
- obydwie 4-kanałowe - odwzorowanie sekcyjno-skojarzeniowe
- zapis pamięci danych - write back
- pamięć L2 512 kB wspólna (kodu i danych) - dostęp po magistrali BSB

Pentium 4

- po 16 kB pamięci L1 danych i kodu
- obydwie 4 kanłowe - asocjacja zespołowa, zorganizowane w 64- bajtowe linijki
- zapis pamięci danych *write trough*
- wspólna pamięć L2 (kodu i danych) 256 kB
- L2 zorganizowana w 128 B bajtowe linijki
- L2 odwzorowanie - asocjcja zespołowa 8-kanałowa