



Politechnika Wrocławska

Struktura pamięci DDR3

na podstawie analizy czasu dostępu do danych

Spis treści

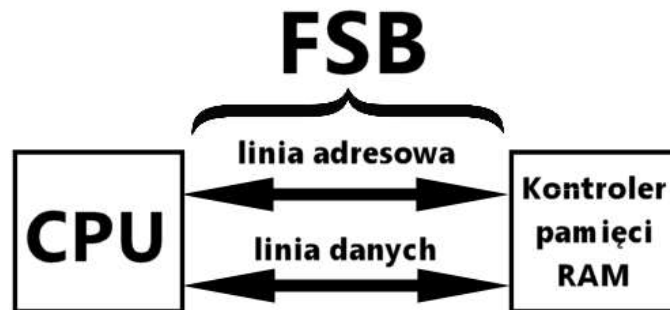
Spis treści	1
Cel projektu	2
Podstawowe pojęcia	2
Wstęp teoretyczny	6
Aplikacja pomiarowa	9
Uruchamianie i obsługa	9
Struktura kodu aplikacji	11
Przebieg badań.....	13
Pamięć podręczna procesora (cache).....	14
Chybienia na stronie (ang. <i>RAS precharge</i>)	15
Przełączanie banków	16
Spostrzeżenia i wnioski.....	17
Bibliografia	18

Cel projektu

Celem projektu jest przeprowadzenie analizy struktury pamięci RAM typu DDR3 od strony wydajnościowej. Zadaniem bazowym jest wykazanie opóźnień w odczycie danych, wynikłych z technologii jej wykonania. Opiera się ona na kontrolowanym wywoływaniu trafień i chybień w poszczególne struktury przechowywania danych w pamięci oraz późniejszej analizie czasu trwania każdego z przypadków.

Podstawowe pojęcia

- **częstotliwość cyklu zegarowego** (*ang. clock rate*) – podstawowa szybkość określona w cyklach na sekundę, z jaką komputer wykonuje podstawowe operacje. Jest określana przez częstotliwość generatora sygnału zegarowego.
- **magistrala systemowa (FSB)** – zespół linii przenoszących sygnały [1] łączące procesor z kontrolerem pamięci RAM. Składają się one z linii adresowej oraz linii danych (Rysunek 1). Przepustowość magistrali systemowej oraz jej częstotliwość taktowania zależy od parametrów i typu zastosowanego procesora.



Rysunek 1. Schemat działania magistrali systemowej FSB

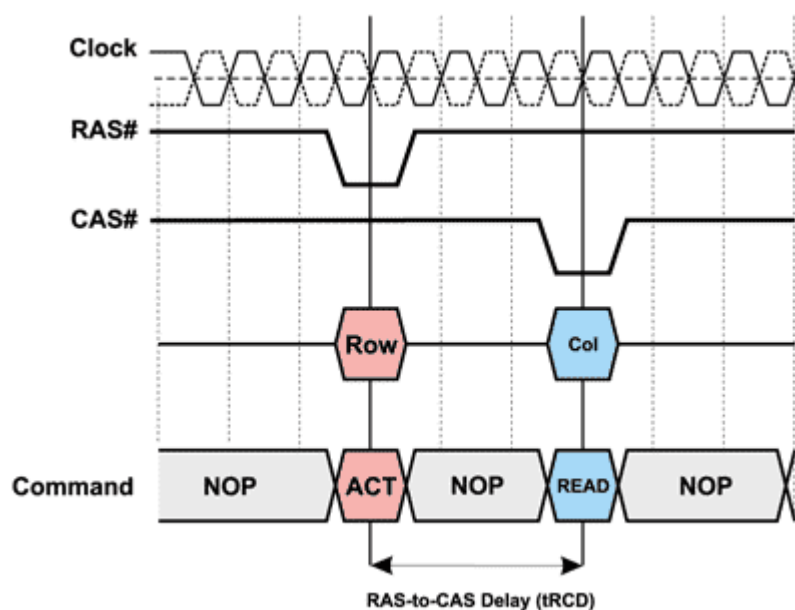
DDR SDRAM (*ang. Double Data Rate Synchronous Dynamic Random Access Memory*) (

- Rysunek 2) –obecnie najpowszechniejszy rodzaj pamięci RAM. W odróżnieniu od jego poprzedników, cechuje się tym, że:
 - dane wymieniane są nie tylko podczas wzrostu zbocza impulsu elektrycznego reprezentującego cykl zegarowy, ale i podczas jego opadania, co podwaja efektywną przepustowość;
 - zegar pamięci jest zsynchronizowany z zegarem magistrali systemowej, dzięki czemu nie jest konieczne oczekiwanie na zgranie się impulsów zegarowych pamięci i FSB ze sobą w celu wykonania żądania [2];
 - jest wykonana z mniejszej ilości tranzystorów niż pamięć statyczna, dlatego też w celu zapobiegania zaniku danych wymaga odświeżania (regeneracji) co kilka cykli.



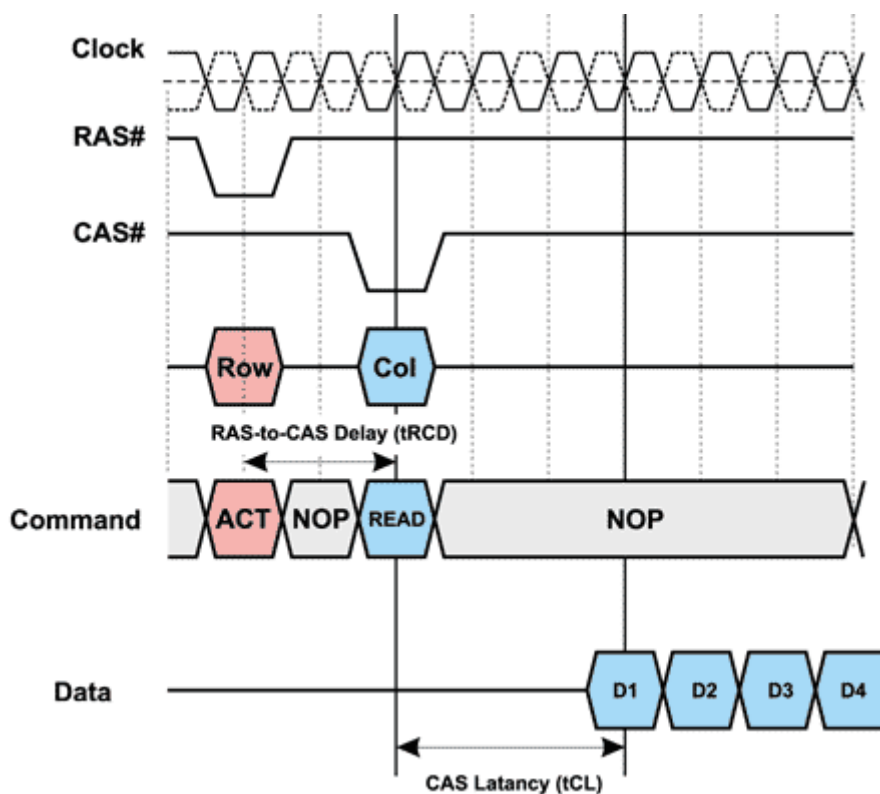
Rysunek 2. Kość pamięci DDR3 oparta na układach firmy Hynix

- **RAS to CAS Delay (RCD)** – opóźnienie w liczbie cykli pomiędzy przestaniem do pamięci numeru wiersza a numeru kolumny (Rysunek 3).



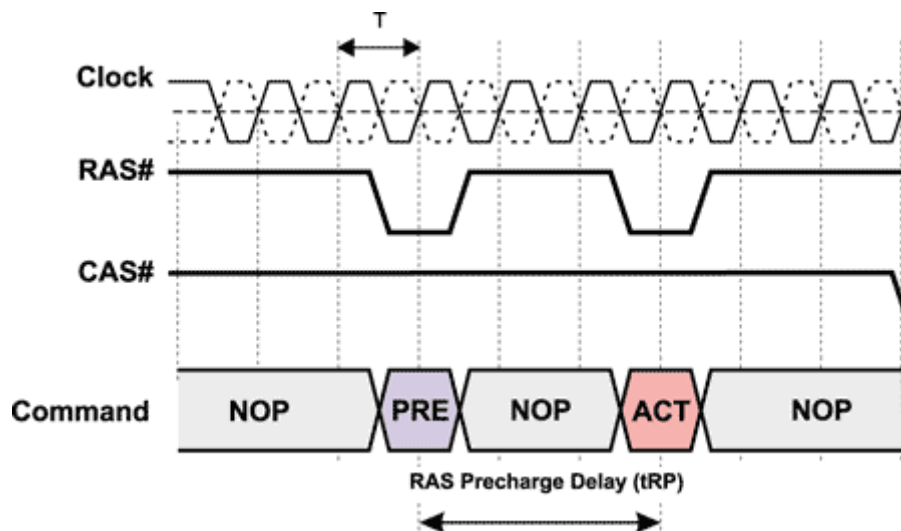
Rysunek 3. Opóźnienie RAS to CAS Delay (t_{RCD}) [3]

- **CAS Latency (CL)** – opóźnienie w liczbie cykli zegara pamięci pomiędzy momentem wystąpienia do niej przez kontroler żądanego numeru kolumny a otrzymaniem danych na wyjściu. Jest to jeden z podstawowych parametrów określających jej szybkość pracy, nierzadko zawarty nawet w nazwie danego modelu pamięci.



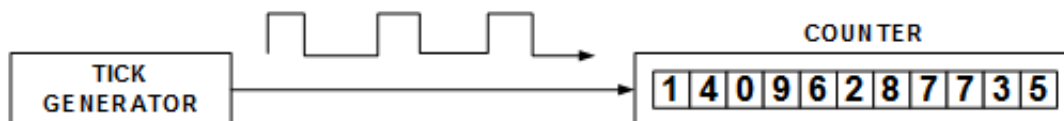
Rysunek 4. Opóźnienie CAS Latency (t_{CL}) [3]

- **RAS precharge (RP)** – opóźnienie w cyklach od wykonania polecenia zamknięcia dostępu do wcześniej aktywowanego wiersza i rozpoczęcia wykonywania polecenia aktywacji kolejnego.



Rysunek 5. Opóźnienie RAS Precharge (t_{RP}) [3]

- **bank pamięci** (*ang. memory bank*) – logiczna jednostka przechowująca dane. Składa się z wierszy i kolumn, można ją więc porównać do macierzy. Rozmiar banku jest zdeterminowany przez ilość bitów adresujących kolumnę i wiersz pomnożone przez ilość chipów jakie bank zawiera [4].
- **Time Stamp Counter (TSC)** – licznik cykli procesora wykonanych od jego resetu [5].



Rysunek 6. Schemat działania licznika ilości wykonanych cykli

- **Pamięć podręczna procesora** (*ang. cache*) – pamięć statyczna typu SRAM o krótkim czasie dostępu, zazwyczaj zlokalizowana bezpośrednio w jądrze procesora. Odgrywa kluczową rolę w przyspieszaniu i upłynnieniu odczytu danych z pamięci RAM, w szczególności w trybie sekwencyjnym.

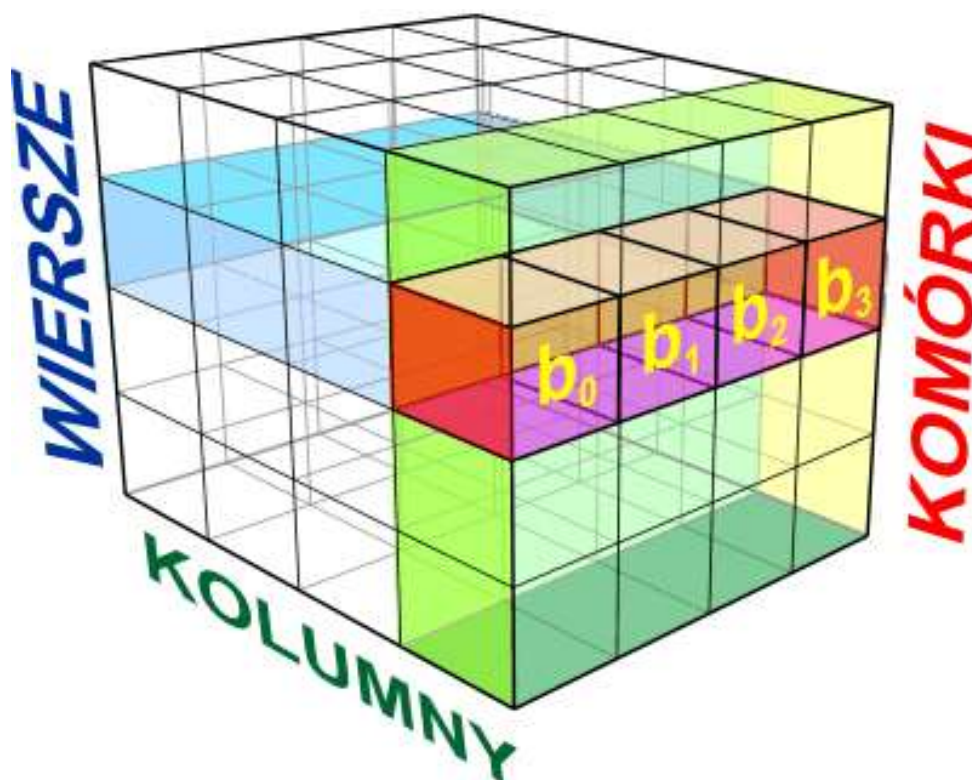
Wstęp teoretyczny

Pamięć RAM to podzespół bazowy komputera. Jej celem jest przechowywanie tymczasowych danych, których rejestry procesora, ze względu na ich ograniczony rozmiar, nie są w stanie pomieścić. Najważniejszą cechą, jaką oferuje, jest błyskawiczny dostęp do informacji.

Zarówno ten, jak i każdy inny element komputera, regularnie ewoluuje. Na przestrzeni 20 lat taktowanie pamięci w przeciętnej klasy domowym komputerze wzrosło z 66 MHz (SDRAM) do 2000 MHz (DDR3, taktowanie efektywne), a przepustowość odpowiednio z 533 MB/s do około 16 GB/s. Postęp może wydawać się kolosalny, jednak na tle ewolucji jaką w tym samym czasie osiągnęły procesory – jest niewystarczający. Szybkość, z jaką ówczesne jednostki centralne mogły wykonywać obliczenia, kilkukrotnie przewyższa możliwości pamięci operacyjnej [6].

Rozróżniane są dwa główne typy pamięci RAM – statyczna oraz dynamiczna. Pierwsza z nich cechuje się tym, że przechowuje zapisane dane tak długo, jak długo jest zasilana. Dynamiczna zaś wymaga okresowego odświeżania w celu podtrzymania przechowywanych danych. Jest to wada mająca wpływ na szybkość, jednak jej budowa jest prostsza, wymaga mniej elementów, a co za tym idzie pamięć jest znacznie tańsza. To właśnie dzięki niskiej, jak na swoją wydajność cenie – zdominowała obecny rynek.

Pamięć RAM magazynuje dane w kilku (zazwyczaj ośmiu) *bankach*. Te zaś składają się z komórek (b_0 , b_1 , b_2 , b_3 na Rysunek 7) adresowanych za pomocą numeru kolumny i numeru wiersza, z których każda przechowuje jeden bajt informacji.



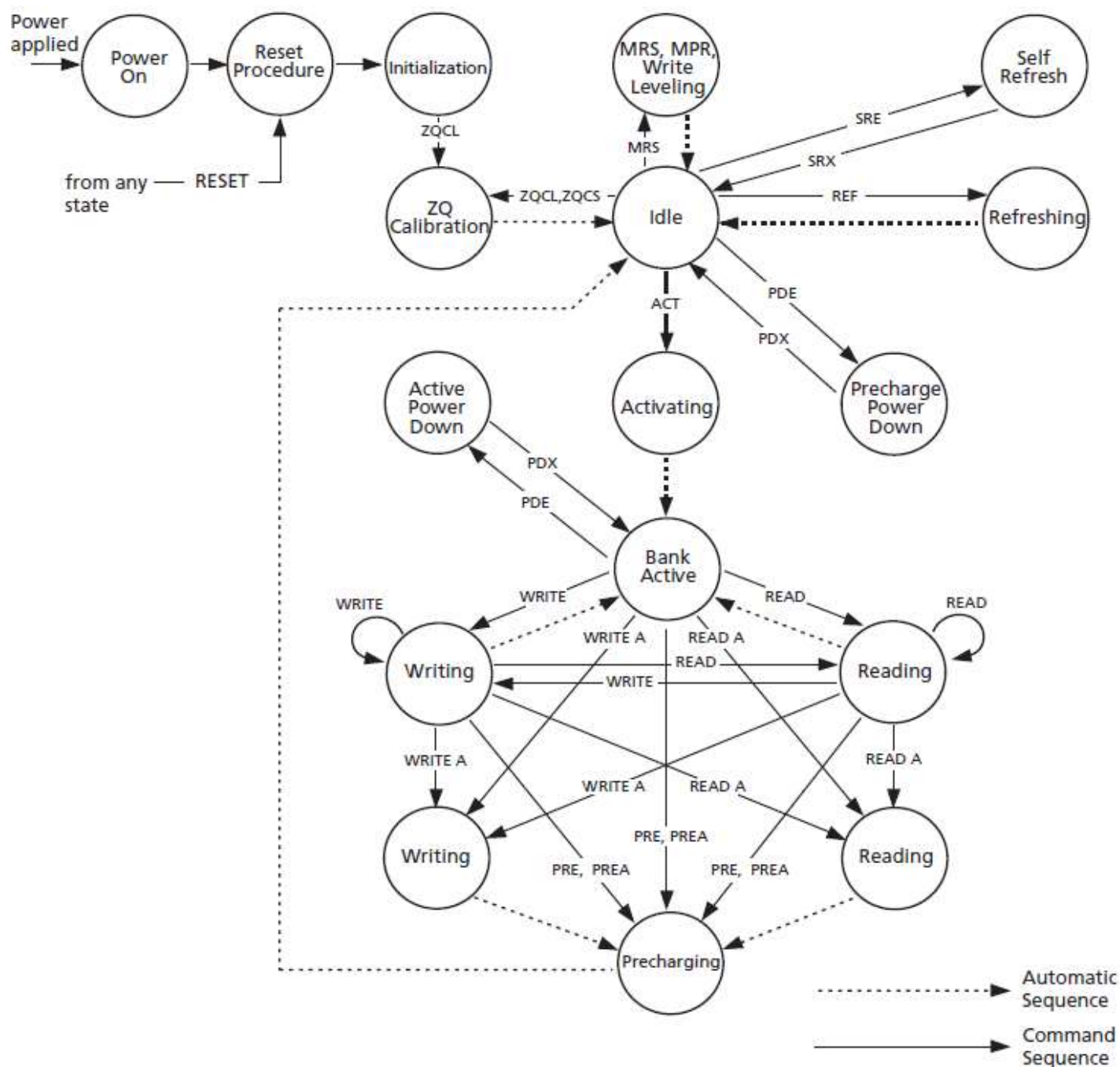
Rysunek 7. Struktura przechowywania danych w banku pamięci [11]

Badaniom poddany został moduł pamięci dynamicznej DDR3 SDRAM firmy *Hynix Semiconductors (Hyundai)* o rozmiarze 4 GB i taktowaniu 800 MHz (efektywnie 1600 MHz). Szczegółowe dane techniczne znajdują się w tabeli poniżej (Tabela 1).

Tabela 1. Dane techniczne modułu Hynix HMT351S6CFR8C-PB uzyskane na podstawie dokumentacji [7]

Dana	Slot #1
Typ pamięci	DDR3
Maksymalne taktowanie (MHz)	800
Maksymalna prędkość transferu (MHz)	1600
Maksymalne Pasma (MB/s)	PC3-12800
Ilość bitów adresujących wiersza	16
Ilość bitów adresujących kolumny	10
Ilość banków	8
Rozmiar banku w megabajtach	512
Szerokość magistrali w bitach	64
Długość odczytu seryjnego (ang. <i>Burst</i>)	8
Minimalny czas cyklu zegara, tCK (ns)	1.250
Minimalny czas CAS Latency (ns)	13.125
Minimalny RAS to CAS Delay, tRCD (ns)	13.125
Minimalny RAS Precharge Time, tRAS (ns)	35.000

Algorytm, przy pomocy którego dany model pamięci RAM dokonuje operacji na danych przedstawić można w postaci automatu (Rysunek 8). Odczytanie informacji o kompletnej sekwencji instrukcji wykonywanych podczas danego żądania staje się wówczas łatwiejsze. Jest to przydatne, gdyż wykonanie każdej z cząstkowych operacji składających się na odczyt danej zajmuje określony czas. Szacowanie jego rzeczywistej wartości uwzględniać powinno więc także takie ewentualności jak kara czasowa spowodowana koniecznością regeneracji banku pamięci. Do wykazania różnic czasowych pomiędzy poszczególnymi metodami dostępu do pamięci, tak dokładne szacowanie nie było jednak konieczne. Pomimo tego, znajomość ewentualnych czynności, jakie pamięć może być zmuszona dodatkowo podjąć przed sięgnięciem do informacji, przydatne było w zrozumieniu nieuzasadnionych wcześniej żadnymi założeniami różnic czasów dostępu.



Rysunek 8. Uproszczony diagram stanów pamięci RAM [7]

Pamięć będąca w stanie beczynności otrzymuje polecenie odczytu wskazanego ciągu komórek danych. Kolejno następują:

- 1) Opcjonalne oczekiwanie na zakończenie procesu automatycznego odświeżania
- 2) Aktywowanie odpowiedniego banku
- 3) Odczyt porcji danych dopóki nie zostanie osiągnięta ostatnia komórka w bieżącym wierszu i/lub nie zostały odczytane wszystkie żądane informacje.
- 4) Wstępne ładowanie (*ang. precharching*) - powrót do punktu 1) celem kontynuowania odczytu z następnego wiersza/banku pamięci.

Aplikacja pomiarowa

Uruchamianie i obsługa

Wychodząc naprzeciw potrzebom badań zaimplementowana została prosta aplikacja. Dostarcza ona narzędzi do analizy czasów pojedynczych instrukcji wykonywanych sekwencyjnie na pamięci oraz przeprowadzania pomiarów czasu masowego odczytu kilkusetmegabajtowej tablicy, z którego następnie wyliczany jest średni czas odczytu pojedynczej komórki.

Środowisko programistyczne, w jakim program był rozwijany to Visual Studio 2013, toteż do kompilacji zalecane jest używanie właśnie tego narzędzia. Aplikacja w postaci plików projektu Visual Studio i kodu źródłowego dostępna jest pod adresem repozytorium *Git*, spod którego projekt należy sklonować do katalogu lokalnego:

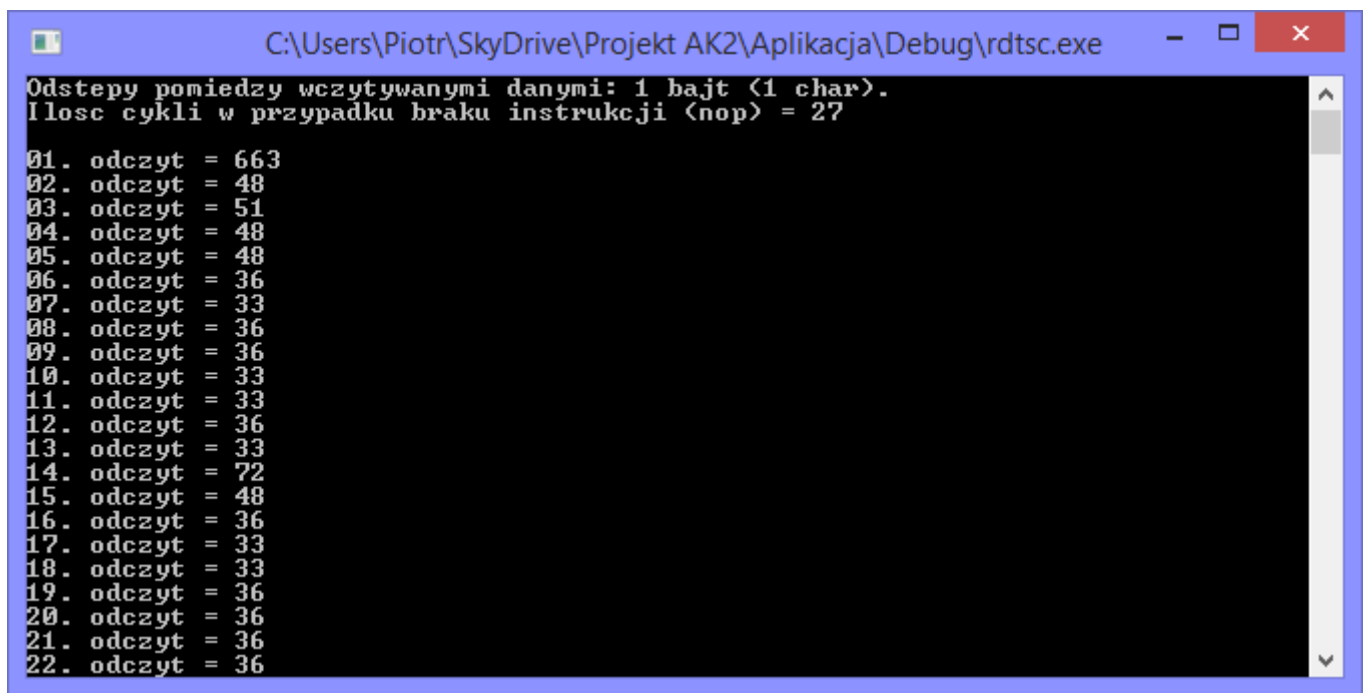
<https://github.com/plawa/DostepDoPamieci>

W celu jej uruchomienia, należy otworzyć projekt zapisany w pliku o rozszerzeniu *vcxproj*, po czym skompilować go. Ze względu na udostępniony kod źródłowy, przed kompilacją możliwa jest edycja dowolnych parametrów mających wpływ na przebieg pomiarów. Parametry mające największy udział w otrzymanych wynikach pomiarów to wielkość skoku, oraz rozmiar zajmowanego przez program w pamięci miejsca. Znajdują się one przed funkcją *main()*, w sekcji kodu określonej komentarzem „****ZMIENNE MODYFIKOWALNE****”.

Podczas uruchamiania aplikacji, w pamięci rezerwowana jest tablica o rozmiarze domyślnie 450 MB, która następnie wypełniana jest losowymi, jednobajtowymi liczbami z przedziału [0; 255]. Do wyboru są następujące metody pomiaru czasu dostępu do pamięci:

1. Preparowanie sekwencji (pomiar masowy). Program pyta użytkownika o wielkość kroku pomiędzy odczytywanymi danymi. Następnie, w oparciu o funkcje z biblioteki *time.h* mierzony jest sumaryczny czas odczytu wszystkich elementów tablicy. Na podstawie uzyskanego wyniku obliczany jest średni czas dostępu do jednego elementu. Dalsze opcje:
 - dowolny klawisz ponawia test¹
2. Pomiar czasu pojedynczej instrukcji (realizowany przy użyciu licznika czasu wysokiej rozdzielczości *TSC*). Zliczana jest ilość cykli procesora, które miną nim otrzymane zostaną żądane informacje. Poprawność wyników gwarantuje nowa instrukcja procesora – *RDTSCP*. W odróżnieniu od jej starszej wersji, nim zostanie wykonana, oczekuje na zakończenie wszystkich poprzednich poleceń [8]. Odczyt powtarzany jest 2050 razy, za każdym pobierając kolejną komórkę ze zdefiniowanej wcześniej tablicy. Po zakończeniu testu, który zwykle trwa ułamek sekundy, rezultaty zapisywane są do pliku o nazwie *wyniki.txt*. Znajduje się w nim lista z odrębnymi wynikami dla każdej z prób. Aplikacja oczekuje na reakcję ze strony użytkownika:
 - przycisk „Enter” podwaja obecny skok pomiędzy wczytywanymi komórkami danych
 - dowolny inny klawisz ponawia test

¹Po przeprowadzeniu pierwszego testu, ze względu na obecność wczytanych danych w pamięci podręcznej procesora, korzystanie z tej opcji jest odradzane i służy jedynie do zobrazowania wpływu cache na prędkość odczytu.



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\Piotr\SkyDrive\Projekt AK2\Aplikacja\Debug\rdtsc.exe. The window contains the following text:

```
Odstępy pomiędzy wczytywanymi danymi: 1 bajt (1 char).  
Ilość cykli w przypadku braku instrukcji (nop) = 27  
  
01. odczyt = 663  
02. odczyt = 48  
03. odczyt = 51  
04. odczyt = 48  
05. odczyt = 48  
06. odczyt = 36  
07. odczyt = 33  
08. odczyt = 36  
09. odczyt = 36  
10. odczyt = 33  
11. odczyt = 33  
12. odczyt = 36  
13. odczyt = 33  
14. odczyt = 72  
15. odczyt = 48  
16. odczyt = 36  
17. odczyt = 33  
18. odczyt = 33  
19. odczyt = 36  
20. odczyt = 36  
21. odczyt = 36  
22. odczyt = 36
```

Rysunek 9. Interfejs programu służącego do pomiaru czasów dostępu do pamięci z wykonanym już pierwszym testem

Struktura kodu aplikacji

Bieżący rozdział zawiera najważniejsze fragmenty kodu źródłowego aplikacji używanej do pomiarów. Wszelkie działania na pamięci oparte zostały o wstawki assemblerowe tak, by zminimalizować liczbę odniesień do niej do niezbędnego minimum. W przypadku mierzenia czasu pojedynczej instrukcji, przesunięcie (*ang. offset*) podczas każdej iteracji wczytywane jest z pamięci do rejestru procesora jeszcze przed rozpoczęciem właściwej fazy pomiaru za pomocą osobnego makra „IterToEDI()”. Ma to na celu zapewnienie maksymalnej izolacji pojedynczego odczytu od pozostałych operacji (Listing 1).

Listing 1. Makra assemblerowe zapewniające dostęp do pamięci na najniższym poziomie abstrakcji

```
//pierwszy odczyt czasu, przeniesienie młodszej części TSC
//do rejestru który nie zostanie nadpisany w drugim etapie pomiaru czasu
#define RDTSCP1() __asm rdtscp \
    __asm mov ebx, eax

//drugi odczyt czasu, odjęcie od nowej wartości TSC wartości
//z poprzedniego odczytu i przekazanie uzyskanego wyniku do zmiennej "x"
#define RDTSCP2(x) __asm rdtscp \
    __asm sub eax, ebx \
    __asm mov x, eax

//wczytanie offsetu tablicy (zmienna "i") do rejestru edi
#define IterToEDI() __asm mov edi, dword ptr[i]

//wykonanie operacji dostępu do komórki pamięci wskazywanej przez adres pod zmienną "tablica"
//oraz index (offset) wczytany do rejestru edi poprzednią funkcją
#define Odczytaj(tab) __asm mov cl, byte ptr[edi + tab]

//pomiar czasów odczytu dla każdej z 2050 kolejnych komórek
unsigned int a, odstep = 1;
for (unsigned int i = 0; i < 2050; i += odstep) {
    IterToEDI(); //wczytanie iteratora do rejestru EDI
    RDTSCP1(); //czas start
    Odczytaj(tablica); //wczytanie bajtu spod adresu "tablica + i" do rejestru cl
    RDTSCP2(a); //czas stop
    fprintf(plik, "%.02d. odczyt = %d\n", i / odstep, a);
}
```

Technika preparowania sekwencji opiera się również na wstawkach assemblerowych, jednak pomiar czasu realizowany jest już nie przy użyciu instrukcji *RDTSCP*, a funkcji biblioteki języka C - *time.h*. Jest ona mniej precyzyjna ze względu na to, że w pomiarze uwzględniany jest też czas odczytu do rejestru iteratora. Przedstawiony kod (Listing 2) demonstruje algorytm pomiaru czasu odczytu całej tablicy w sposób sekwencyjny, przy założeniu, że wielkość skoku równa się jednemu bajtowi.

Listing 2. Funkcje wykorzystywane do mierzenia czasu techniką preparowania sekwencji oraz algorytm pomiaru

```
//ustawia bieżący czas podawanemu parametrowi
void startCzas(clock_t *startTime){
    *startTime = clock();
}

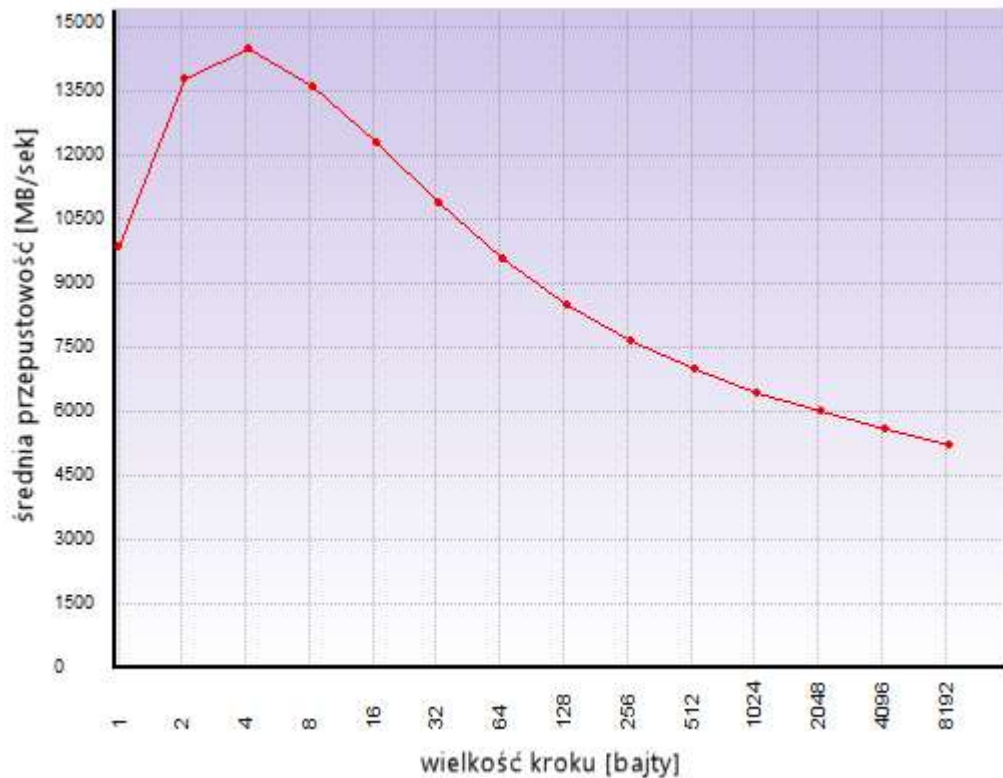
//na podstawie aktualnego i przekazanego jako parametr
//wcześniejszego czasu - oblicza czas, jaki upłynął
double stopCzas(clock_t *startTime){
    return static_cast <double>(clock() - *startTime) / CLOCKS_PER_SEC;
}

//pomiar czasu odczytu całej tablicy wraz z obliczeniem średniego czasu odczytu pojedynczej komórki
startCzas(czasStartu);
for (int i = 0; i < iloscElemDoOdczytu; i++){
    IterToEDI();
    Odczytaj(tablica);
}
zmierzonyCzas = stopCzas(czasStartu);
printf("Czas odczytu %d elementow z pamieci: %.3f [s]\n", iloscElemDoOdczytu, zmierzonyCzas);
```

```
printf("Sredni czas odczytu dla jednej komórki: %.3f[ns]",1000000000*zmierzonyCzas/iloscElemDoOdczytu);
```

Przebieg badań

Przed przystąpieniem do właściwej fazy testów, przeprowadzono test prędkości odczytu z pamięci za pomocą ogólnodostępnego narzędzia firmy *PassMark* – *PerformanceTest*. Jako kryterium zależności wybrana została wielkość skoku pomiędzy kolejnymi odczytywanymi komórkami w pamięci. Wyniki jednoznacznie wykazały, iż optymalną wartością kroku są 4 bajty (Rysunek 10). Dzieje się tak, ponieważ procesor, płyta główna oraz pamięć RAM są zoptymalizowane pod odczyty porcji danych o długości 32 bitów. Dłuższy krok minimalizuje korzyści, jakie dostarcza pamięć podręczna procesora. Oznacza także częstszą konieczność wykonywania czasochłonnych operacji takich jak przestanie numeru nowego wiersza, czy zmiana banku.



Rysunek 10. Wykres zależności średniej prędkości odczytu danych od wielkości skoku pomiędzy komórkami

Badania skoncentrowane zostały na pomiarze sekwencyjnych odczytów o kroku równym jeden. Analiza wykresu zawartego na Rysunek 10 pozwoliła stwierdzić, iż pamięć nie będzie mierzona przy parametrach gwarantujących najszybszy odczyt, jednak metoda ta jest mniej skomplikowana i pozwala na lepsze zobrazowanie struktury pamięci RAM.

Przeprowadzony eksperyment skupiony został na wykryciu tzw. „wąskich gardeł” (*ang. „bottleneck”*) pamięci, czyli opóźnień spowodowanych:

- ograniczoną wielkością pamięci podręcznej (*cache*) procesora
- chybieniami na stronie
- przełączaniem banków

Pamięć podręczna procesora (cache)

Pierwszym krokiem był pomiar czasów odczytu poszczególnych bajtów w pamięci. Podczas wykonywania testu zauważona została pewna, niewynikająca z parametrów technicznych pamięci prawidłowość. Otóż opóźnienie w dostępie do danych z wyjątkiem tych odczytywanych co 64 bajty jest praktycznie niezauważalne (ich czas porównać można do czasu wykonywania polecenia *nop*, czyli po prostu braku operacji). Uzyskanie odpowiedzi wyjaśniającej taki stan rzeczy wymagało sięgnięcia do specyfikacji technicznej procesora *Intel i7 3612 QM* [9]. Wynika z niej, iż pamięć podręczna owej jednostki centralnej posiada właśnie 64 bajtową linię. Z każdym sięgnięciem do pamięci, pobierany jest więc nie jeden, a aż 64 bajty informacji, które przekazywane są do cache. Sytuacja ta powoduje, że kolejne 63 jednobajtowe komórki zamiast z pamięci RAM pobierane są już z pamięci podręcznej procesora, co znacząco poprawia szybkość odczytu. Taką metodę odczytu danych nazywa się seryjnym trybem odczytu (ang. *Burst Mode*) [10].

Rysunek 11. Fragment wyniku działania programu ilustrujący wpływ pamięci podręcznej procesora na szybkość odczytu.

```
63. odczyt = 33
64. odczyt = 180 ← odczyt z pamięci RAM
65. odczyt = 36
66. odczyt = 33
67. odczyt = 33 ← odczyt z cache
.
.
.
.
.
124. odczyt = 33
125. odczyt = 33
126. odczyt = 45
127. odczyt = 48 ← odczyt ostatniego bajtu z cache'u
128. odczyt = 189 ← ponowny odczyt z pamięci RAM
129. odczyt = 33
```

Następnym krokiem było wykonanie pomiaru przy użyciu drugiej z metod (opisana w rozdziale *Aplikacja pomiarowa*, podrozdział *Uruchamianie i obsługa*, punkt 2). Ze względu na wspomaganie cache, czas ten był składową czasów odczytu danych z pamięci RAM oraz czasów odczytu danych z pamięci podręcznej procesora. Średnia z 10 prób wyniosła **2,417 ns** przy odchyleniu standardowym równym **0,249 ns**.

W celu wyeliminowania udziału cache procesora przy odczycie danych, w następnym pomiarze krok odczytu został ustawiony na długość linii bufora pamięci podręcznej. W znacznym stopniu wydłużyło to średni czas odczytu pojedynczej komórki, bo aż do **7,067 ns** (średnia z 10 prób, odchylenie standardowe równe **1,215 ns**).

Chybiecie na wierszu (opóźnienie RAS precharge)

Przez termin ten określa się przypadek, gdy żądana komórka znajduje się w innym wierszu aniżeli aktualnie otwartym. Wiąże się z tym potrzeba przesłania dodatkowo numeru wiersza (zamiast jedynie numeru kolumny), co wydłuża czas trwania operacji. Z danych technicznych badanego modułu (

Tabela 1) wynika, że czas ten jest ponad trzykrotnie dłuższy niż w przypadku odczytu danych z otwartego już wcześniej wiersza.

Wykazanie tej prawidłowości sprowadzało się do odczytania większej ilości informacji niż jest w stanie zmieścić się w jednym wierszu oraz sprawdzeniu czasu odczytu komórki będącej początkiem nowego wiersza. Z dokumentacji modułu wynika, że posiada on 10 bitów adresujących kolumnę. Za pomocą 10 bitów można zakodować 1024 różnych liczb, zatem tyle też wynosi długość wiersza. Rachując w analogiczny sposób, dla 16 bitów adresujących wiersze - ilość wierszy wynosi 65536.

Przy użyciu opisanej w niniejszym sprawozdaniu aplikacji, przeprowadzony został test, który potwierdził zgodność z danymi technicznymi. Wyraźnie widać, że zamknięcie aktualnego wiersza i otwarcie nowego, jest ponad trzykrotnie bardziej czasochłonne niż odczyt danych na otwartej stronie (Rysunek 12).

```
960. odczyt = 186 ← odczyt z pamięci RAM
.
.
.
1023. odczyt = 30 ← odczyt z cache'u
1024. odczyt = 708 ← zmiana wiersza i odczyt z pamięci RAM
1025. odczyt = 27
1026. odczyt = 30
1027. odczyt = 27
```

Rysunek 12. Fragment wyniku działania programu ilustrujący opóźnienie wynikłe z konieczności zmiany wiersza.

Średni czas odczytu jednej komórki uzyskany na podstawie 10 prób przeprowadzonych za pomocą techniki preparowania sekwencji w programie wyniósł **26,791 ns**. Odchylenie standardowe o wartości **4,574 ns** wskazuje, że odczyty różniły się między sobą dość znacznie. Minimalny zmierzony czas wyniósł **20,345 ns**, a maksymalny - **36,621 ns**.

```
Podaj odstep miedzy komorkami: 1024
Czas odczytu 491520 elementow z pamieci wynosi: 0.015 [s]
Średni czas odczytu dla jednej komorki danych wynosi: 30.518 [ns]
```

Rysunek 13. Jedna z prób masowego pomiaru czasu odczytu w sytuacji chybień na stronie

Przełączanie banków

Ostatnim, wyraźnie zauważalnym opóźnieniem w odczycie danych z pamięci RAM, jest zaistnienie konieczności przełączenia banku. Potrzeba ta występuje w sytuacji, gdy żądane komórki z uwagi na niewystarczającą ilość pamięci nie mogły zostać zapisane w tym samym banku co reszta informacji. Jest to pewnego rodzaju fragmentacja danych.

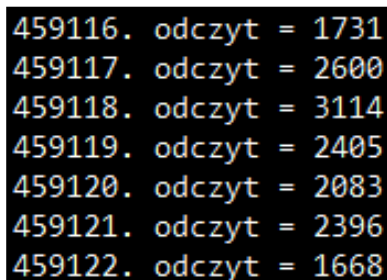
Testowany moduł zbudowany jest z ośmiu banków, po 512 MB każdy - daje to sumaryczną pojemność 4 GB. W celu zaobserwowania momentu w którym bank zostaje przełączony, modyfikacji uległ kod źródłowy programu testującego. Do pamięci programu została wczytana druga 450 MB tablica zmiennych typu *char*, a procedura mierzenia zmodyfikowana tak, by podczas każdej z iteracji odczyt następował naprzemiennie z jednej i z drugiej tablicy (Listing 3). W ten sposób procesor zmuszony został do sekwencyjnego przełączania banków.

Listing 3. Procedura odczytu wymuszająca przełączanie banków pamięci

```
for (unsigned int i = 0; i < rozmiarAlokByte; i+= skok){  
    IterToEDI();           //wczytanie iteratora i (offsetu tablicy) do rejestru edi  
    Odczytaj(tablica);     //odczyt z pierwszej tablicy  
    Odczytaj(tablica2);    //odczyt z drugiej tablicy  
}
```

Przyniosło to spodziewane efekty, bowiem zanotowane opóźnienia są średnio trzykrotnie wyższe niż w przypadku chybienia na stronie (

Rysunek 14).



```
459116. odczyt = 1731  
459117. odczyt = 2600  
459118. odczyt = 3114  
459119. odczyt = 2405  
459120. odczyt = 2083  
459121. odczyt = 2396  
459122. odczyt = 1668
```

Rysunek 14. Fragment wyniku działania programu ilustrujący opóźnienie wynikłe z konieczności zmiany banku.

Następnym krokiem było wykonanie pomiarów przy użyciu techniki preparowania sekwencji. Ze względu na rozbieżności pomiędzy wynikami, test, podobnie jak w poprzednim przypadku, został przeprowadzony również 10-krotnie. Średni czas odczytu jednej komórki pamięci dla wszystkich prób wyniósł aż **72,123 ns**, a odchylenie standardowe kształtowało się na poziomie 3,032 ns. Najkrótszy średni czas odczyt spośród wszystkich prób wyniósł 67,139 ns, a najdłuższy – 75,277 ns.

Spostrzeżenia i wnioski

- Wyniki badań w najprostszy sposób przedstawić można za pomocą tabeli obrazującej strukturę ułożenia danych w banku. Kolory odpowiadają poszczególnym prędkościom odczytu przy założeniu, że dane odczytywane są sekwencyjnie jedna po drugiej począwszy od komórki o numerze 0. Symbolicznie czarnym kolorem oznaczony został koniec banku, co wiąże się z koniecznością aktywowania następnego (proces ten trwa najdłużej spośród wszystkich obecnie wymienionych). Kolor czerwony oznacza zmianę wiersza (trwa nieco krócej niż zmiana banku), kolor żółty – odczyt z pamięci, a kolor zielony – odczyt z pamięci podręcznej procesora (trwa najkrócej).

Tabela 2. Struktura banku pamięci wraz z odpowiadającym poszczególnym komórkom prędkościom odczytu (przy założeniu, że odczyt następuje sekwencyjnie począwszy od komórki o numerze 0).

		numer kolumny											
nr wiersza	0	1	2	...	63	64	65	...	127	128	129	...	1023
	1024	1025	1026	...	1087	1088	1089	...	1151	1152	1153	...	2047
	...												
	65535												
następny bank													

- Wyniki badań w większości pokryły się z parametrami odczytanymi ze specyfikacji badanego modułu pamięci.
- Wiele komplikacji sprawiał fakt, iż na maszynie testującej zainstalowane były dwa moduły pamięci współpracujące ze sobą w trybie *Dual Channel*. Celem rozwiązania problemu, z komputera wyjęta została jedna kość. Nieregularności w czasach odczytu dla poszczególnych komórek nadal jednak nie zniknęły. Przyczyną takiego stanu rzeczy najprawdopodobniej był brak izolacji środowiska testowego od środowiska systemu operacyjnego wraz z usługami działającymi w tle. Również struktury danych mogły zostać umieszczone w pamięci nierównomiernie.
- Pamięć RAM, pomimo tego iż zapewnia dużą prędkość przetwarzania danych, ma też swoje słabe strony. Ich znajomość umożliwia programistom opracowywanie algorytmów w taki sposób, aby wynikiem ich pracy były programy o maksymalnej możliwej wydajności. Przykładowo programista, wiedząc, że szybkość działania jest priorytetem w pisanej przez niego aplikacji, zwróci uwagę na fakt, by dane, które zazwyczaj będą odczytywane sekwencyjnie, nie były dzielone pomiędzy wierszami. Natomiast w przypadku aplikacji, które wymagają więcej pamięci RAM niż jest w stanie pomieścić jeden jej bank – zapewni, by częściej używane dane nie zostały pofragmentowane po różnych bankach, co mogłoby przyczynić się do istotnego spadku szybkości.
- Pamięć podręczna procesora ma ogromny wpływ na zwiększenie szybkości odczytu informacji z pamięci RAM. Te same dane, dla których zachodzi potrzeba wielokrotnego odczytywania – z kolejną próbą dostępne są coraz szybciej i szybciej (Rysunek 15). Dzieje się tak, ponieważ dane w pamięci podręcznej nie są usuwane – są nadpisywane. Jeżeli inna aplikacja działająca w tle ich nie naruszy, ponowne sięganie do pamięci RAM po te same dane zostanie ograniczone do minimum. Rezultatem będzie kilkukrotnie krótszy czas dostępu do danych.

```
Czas odczytu 7500000 elementów z pamięci wynosi: 0.014 [s]
Średni czas odczytu dla jednej komórki danych wynosi: 29.867 [ns]
Czas odczytu 7500000 elementów z pamięci wynosi: 0.008 [s]
Średni czas odczytu dla jednej komórki danych wynosi: 17.067 [ns]
Czas odczytu 7500000 elementów z pamięci wynosi: 0.004 [s]
Średni czas odczytu dla jednej komórki danych wynosi: 8.533 [ns]
```

Rysunek 15. Wpływ pamięci podręcznej na szybkość odczytu danych z pamięci RAM

Bibliografia

- [1] „Magistrala komunikacyjna - Wikipedia, wolna encyklopedia,” [Online]. Dostępny: http://pl.wikipedia.org/wiki/Magistrala_komunikacyjna. [Data uzyskania dostępu: 1 Czerwiec 2015].
- [2] „SDRAM,” Wikipedia, wolna encyklopedia, [Online]. Dostępny: <http://pl.wikipedia.org/wiki/SDRAM>. [Data uzyskania dostępu: 26 Maj 2015].
- [3] КомпьютерПресс, „Энциклопедия современной памяти | КомпьютерПресс,” [Online]. Dostępny: <http://compress.ru/article.aspx?id=16737>. [Data uzyskania dostępu: 1 Czerwiec 2015].
- [4] „Memory bank - Wikipedia, the free encyclopedia,” [Online]. Dostępny: http://en.wikipedia.org/wiki/Memory_bank. [Data uzyskania dostępu: 21 Maj 2015].
- [5] „Time Stamp Counter - Wikipedia, the free encyclopedia,” [Online]. Dostępny: http://en.wikipedia.org/wiki/Time_Stamp_Counter. [Data uzyskania dostępu: 21 Maj 2015].
- [6] K. Kaspersky, „Podsystem pamięci RAM - wstęp,” w *Optymalizacja kodu*, Warszawa, RM, 2005, pp. 73-74.
- [7] Hynix, „SK Hynix,” 10 Wrzesień 2012. [Online]. Dostępny: <https://www.skhynix.com/products/computing/view.jsp?info.ramKind=20&info.serialNo=HMT351S6CFR8C>. [Data uzyskania dostępu: 23 Maj 2015].
- [8] Intel Corporation, „RDTSCP—Read Time-Stamp Counter and Processor ID,” w *Intel® 64 and IA-32 Architectures Volume 2: Instruction Set Reference*, 2015, pp. Vol. 2B 4-303.
- [9] Intel Corporation, „Intel® Core™ i7-3612QM Processor,” [Online]. Dostępny: http://ark.intel.com/pl/products/64901/Intel-Core-i7-3612QM-Processor-6M-Cache-up-to-3_10-GHz-BGA. [Data uzyskania dostępu: 1 Czerwiec 2015].
- [10] A. Nowak, „PAMIĘCI PÓŁPRZEWODNIKOWE,” [Online]. Dostępny: http://www.andrzej-nowak.cba.pl/materialy/utk/rozdzial2/pamieci_4.pdf. [Data uzyskania dostępu: 25 Maj 2015].
- [11] J. Walaszek, „Bit w zastosowaniach - edu.i-lo.tarnow.pl,” [Online]. Dostępny: http://edu.i-lo.tarnow.pl/inf/alg/002_struct/0043.php. [Data uzyskania dostępu: 29 Maj 2015].