Systemy operacyjne

Struktura i zasady budowy

Opracował: Andrzej Nowak

Rozdział 1

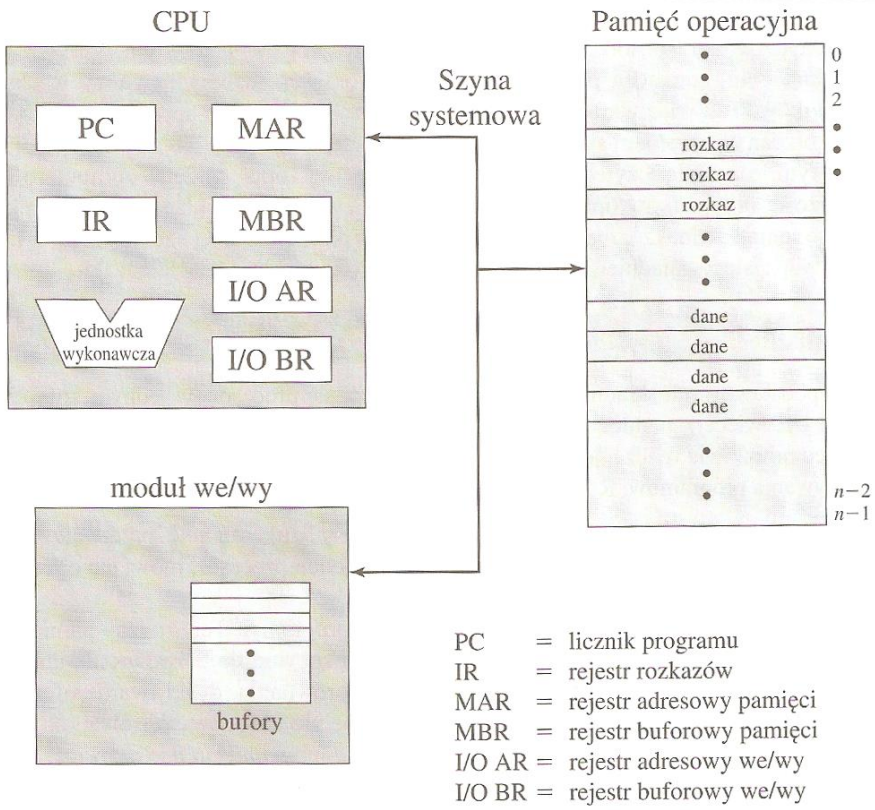
Wprowadzenie do systemów komputerowych

Zadaniem systemu operacyjnego jest pośredniczenie pomiędzy aplikacjami, programami narzędziowymi i użytkownikami a sprzętem komputerowym.

System operacyjny korzysta z zasobów sprzętowych jednego lub kilku procesorów, udostępniając w ten sposób zestaw usług użytkownikom systemowym. Oprócz tego system operacyjny zarządza pamięcią pomocniczą oraz urządzeniami we/wy w imieniu użytkowników.

1.1 Podstawowe komponenty komputera

Procesor – steruje działaniem komputera oraz realizuje funkcje przetwarzania danych. Jeśli w komputerze jest zainstalowany tylko jeden procesor, mówi się o jednostce centralnej (Central processing unit – CPU).
Pamięć operacyjna – przechowuje dane i programy. Nazywana jest pamięcią główną lub pamięcią podstawową.
Moduły wejścia / wyjścia – przemieszczają dane między komputerem a zewnętrznym środowiskiem.
Szyna systemowa – umożliwia komunikację pomiędzy procesorami, pamięcią operacyjną a modułami we/wy.



1.2 Rejestry procesora

W procesorach znajdują się rejestry udostępniające szybszą pamięć o mniejszej pojemności w porównaniu z pamięcią operacyjną .
Rejestry w procesorze wykonują dwa zadania:

1. **Rejestry widoczne dla użytkownika –** ułatwiają projektantom oraz programistom korzystającym z języka asemblerowego minimalizowanie odwołań do pamięci poprzez zoptymalizowanie wykorzystania rejestru. Procesor może odwoływać się do rejestru widocznego dla użytkownika za pomocą języka komputerowego, który procesor wykonuje. Taki rejestr jest ogólnie dostępny dla wszystkich programów, włączając w to aplikacje i programy systemowe.
   Najczęściej dostępnymi rodzajami rejestrów są:
   a. **Rejestry danych** – są to rejestry ogólnego przeznaczenia, co oznacza, że może z nich korzystać dowolna instrukcja przetwarzająca dane. Jednakże, często ich zastosowanie wiąże się z pewnymi ograniczeniami.
   b. **Rejestry adresowe** – zawierają adresy danych lub rozkazów w pamięci operacyjnej lub fragment adresu używanego do obliczenia adresu całkowitego lub efektywnego.
   c. **Wskaźnik segmentu** – pamięć jest dzielona na segmenty, które są po prostu blokami słów o zmiennej długości. Odwołanie do pamięci składa się z odwołania do określonego segmentu oraz przesunięcia wewnątrz tego segmentu.
   d. **Wskaźnik stosu** – jeśli mamy do czynienia z adresowaniem stosu widocznym dla użytkownika, to tworzony jest dedykowany rejestr wskazujący lokalizację na szczycie stosu. Dzięki temu można stosować rozkazy nie zawierające pól adresowych, takie jak push i pop.

2. **Rejestry sterowania i stanu**- wykorzystywane przez procesor w celu sterowania działaniem procesora oraz sterowania wykorzystania programów za pomocą uprzywilejowanych procedur systemu operacyjnego.
   Do kontrolowania działania procesorów służy wiele różnych rejestrów procesora. W większości komputerów rejestry procesora nie są widoczne dla użytkownika. Do niektórych dostęp mogą mieć instrukcje komputera – tryb sterowania lub tryb systemu operacyjnego.
   Rejestry niezbędne podczas wykonywania rozkazów:
      a. **Licznik programu (PC)** – zawiera adres kolejnego rozkazu do pobrania.
      b. **Rejestr rozkazów (IR)** – zawiera ostatnio pobrany rozkaz.

   Inne rejestry:

   MAR – rejestr adresowy pamięci, MBR – rejestr buforowy pamięci, AC - akumulator

   Kody warunków (flagi) – są zazwyczaj bitami ustawianymi przez osprzęt procesora w rezultacie wykonywania zadań. Przykładowo, działanie arytmetyczne może dać wynik dodatni, ujemny, zerowy lub doprowadzić do przepełnienia.

   Oprócz przechowywania wyniku w rejestrze lub w pamięci, kod warunku jest również ustawiany po zakończeniu działania arytmetycznego. Bity kodu warunków są przechowywane w jednym lub kilku rejestrach. Zazwyczaj tworzą część rejestru sterującego. Instrukcje komputera umożliwiają odczytywanie tych bitów za pomocą ukrytych odwołań.

1.3 Wykonywanie rozkazów

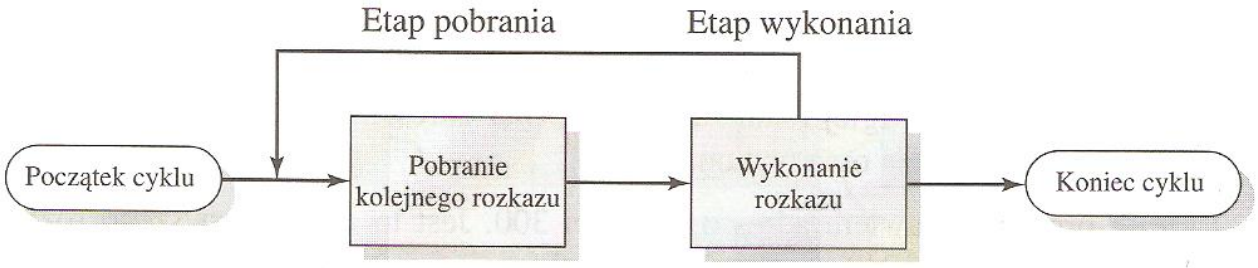
Program , który ma być wykonany przez procesor, składa się z zestawu rozkazów zapisanych w pamięci.
Przetwarzanie rozkazów obejmuje dwa etapy:

a. **Etap pobrania** - procesor pobiera po jednym rozkazie z pamięci,
b. **Etap wykonania** - wykonanie rozkazu pobranego przez procesor .

Wykonywanie programu sprowadza się do powtarzania sekwencji – pobierania rozkazów i ich wykonywania.

Wykonywanie rozkazów może się wiązać z kilkoma czynnościami i jest zależne od rodzaju rozkazu.

Cykl rozkazu - wszystkie operacje związane z przetwarzaniem jednego rozkazu.



Wstrzymanie wykonania programu – może nastąpić , jeśli :

a. Wyłączymy komputer.
b. Pojawią się jakieś poważne błędy.
c. Zostanie wysłany rozkaz wstrzymujący pracę procesora.

1. Pobieranie i wykonywanie rozkazu

W początkowej fazie każdego cyklu procesor pobiera jeden rozkaz z pamięci. W typowym procesorze licznik programu (PC) przechowuje adres kolejnego rozkazu, który ma zostać pobrany. O ile nie ma innych dyrektyw, procesor zawsze zwiększa wartość licznika po każdej operacji pobrania (chodzi o to, aby można było pobrać kolejny rozkaz zlokalizowany nieco wyżej w pamięci).

Pobrany rozkaz ładowany jest do rejestru rozkazu (IR). W rozkazie zawarte są bity określające operację, którą procesor ma wykonać. Procesor interpretuje rozkaz i wykonuje żądaną operację.

Rodzaje operacji:

a. **Procesor–pamięć** – dane można przenosić z procesora do pamięci lub z pamięci do procesora.
b. **Procesor-we/wy** – dane mogą być przenoszone pomiędzy procesorem a modułem we/wy (czyli do lub z urządzenia peryferyjnego).
c. **Przetwarzanie danych** – procesor może realizować na danych działania arytmetyczne lub logiczne.
d. **Sterowanie** – rozkaz może zawierać dyrektywę określającą zmianę kolejności wykonania rozkazu.

Wykonanie rozkazu może obejmować połączenie kilku z wymienionych operacji.

2. Funkcja we/wy

Dane mogą być wymieniane bezpośrednio pomiędzy modułem we/wy (przykładowo kontrolerem dysku) a procesorem. Procesor nie tylko może zainicjować operację odczytu lub zapisu do/z pamięci, ale może również odczytać dane z modułu we/wy lub zapisać je do niego.

W niektórych przypadkach, warto dopuścić do bezpośredniej wymiany danych pomiędzy urządzeniami we/wy a pamięcią, by zwolnić procesor z konieczności komunikowania się z urządzeniami we/wy. Pozwala to na wymianę danych bez angażowania procesora – tzw. bezpośredni dostęp do pamięci realizowany przez układ DMA(Direct Memory Access).

1.4 Przerwania

Praktycznie wszystkie komputery są wyposażone w mechanizm, który umożliwia modułom (takim jak we/wy czy pamięć) przerywanie normalnej sekwencji przetwarzania rozkazów.

Tabela: Klasy przerwań

Program	Wygenerowane przez warunek będący wynikiem wykonania rozkazu. Przykładowo może to być przepełnienie arytmetyczne, dzielenie przez zero, próba wykonania niedozwolonego rozkazu oraz odwołania się poza zakres przestrzeni adresowej dozwolonej dla danego użytkownika.
Czasomierz	Wygenerowane przez czasomierz procesora. Dzięki temu system operacyjny może regularnie realizować wybrane funkcje.
we/wy	Wygenerowane przez kontroler we/wy, by zasygnalizować normalne zakończenie działania lub ewentualne błędy.

Awaria sprzętu	Wygenerowane na skutek awarii sprzętu, do którego może dojść z uwagi na przerwę w zasilaniu lub błąd parzystości pamięci.

Programy we/wy składają się z trzech części:

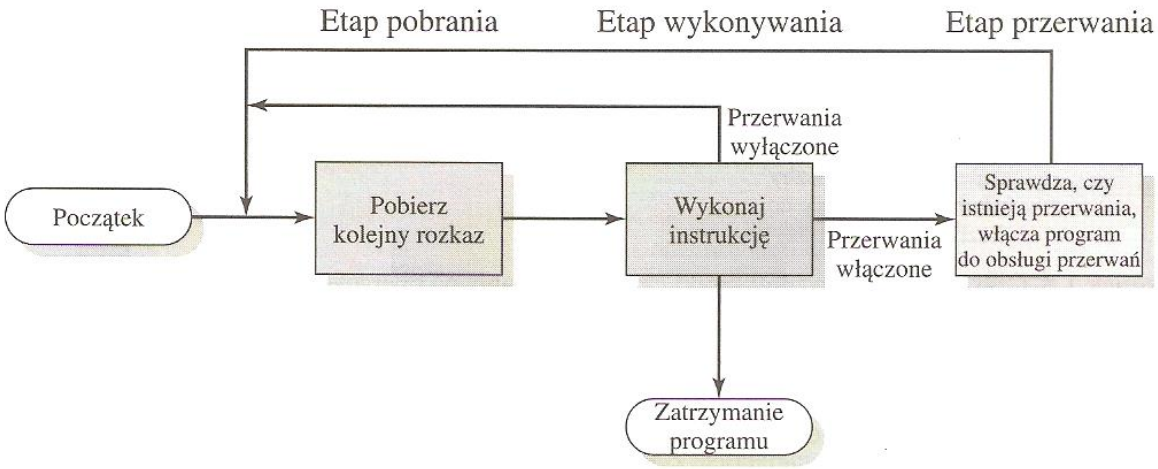
Sekwencja rozkazów przygotowujących system do realizacji faktycznej operacji we/wy - operacja taka może obejmować przekopiowanie danych do specjalnego bufora oraz przygotowanie parametrów polecenia komunikującego się z urządzeniami we/wy.

Rzeczywiste polecenie we/wy – w przypadku braku przerwań po wywołaniu polecenia program musi zaczekać, aż urządzenie we/wy wykona zażądaną operację (lub od czasu do czasu musi sprawdzać status urządzenia we/wy).

Sekwencja rozkazów potrzebnych do zakończenia operacji – przykładowe rozkazy to ustawienie flagi pomyślnego zakończenia operacji bądź wystąpienia błędu.

Przerwania a cykl rozkazów

Dzięki przerwaniom procesor może się zająć wykonywaniem innych rozkazów podczas realizacji operacji we/wy.



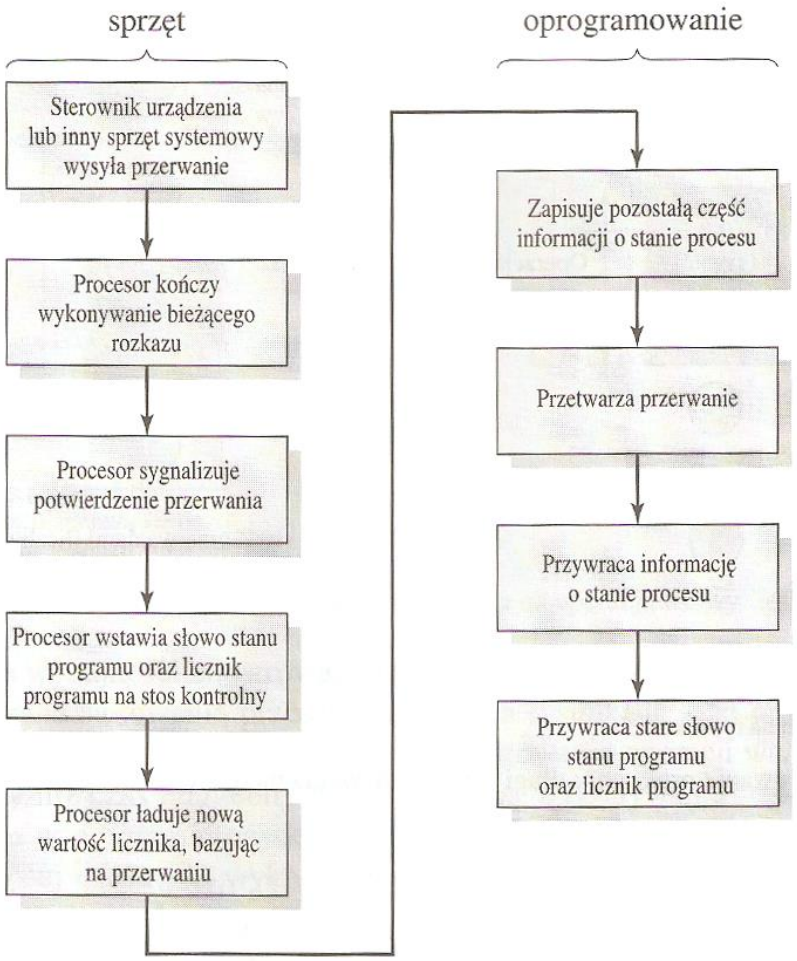
Aby obsłużyć przerwanie, do cyklu rozkazów dodaje się **etap przerwania**. Na tym etapie procesor sprawdza, czy nie pojawiły się jakieś przerwania zgłaszane za pomocą sygnału przerwania.

- Jeśli nie ma żadnych przerwań, procesor przechodzi do etapu pobrania i pobiera kolejną instrukcję programu.
- Jeśli jakieś przerwanie zostało zgłoszone, procesor wstrzymuje wykonywanie bieżącego programu i wykonuje **procedurę obsługi przerwań**. Procedura ta jest zazwyczaj częścią systemu operacyjnego. Procedura sprawdza rodzaj przerwania i wykonuje potrzebne działanie.

Kiedy procedura obsługi przerwań kończy działanie, procesor może wznowić wykonywanie programu użytkownika, który został zatrzymany w punkcie przerwania.

Przetwarzanie przerwań

Wystąpienie przerwania wyzwala pewną liczbę zdarzeń zarówno w osprzęcie jak i w oprogramowaniu procesora.



Kiedy urządzenie we/wy zakończy operację we/wy, następujące zdarzenia mają miejsce na warstwie sprzętowej:

1. Urządzenie wysyła sygnał przerwania do procesora.
2. Procesor kończy wykonywanie bieżącego rozkazu przed obsłużeniem przerwania.
3. Procesor sprawdza, czy nie pojawiło się żądanie przerwania, określa, że takie istnieje, po czym wysyła sygnał potwierdzenia do urządzenia, które wysłało żądanie przerwania. To potwierdzenie umożliwia urządzeniu usunięcie sygnału przerwania.
4. Procesor musi teraz przygotować transfer sterowania do procedury obsługi przerwań.
    - Najpierw konieczne jest zapisanie informacji potrzebnych do wznowienia bieżącego programu od punktu przerwania.
    - Potrzebne będzie słowo stanu programu (PSW) oraz lokalizacja następnego rozkazu, który ma zostać wykonany.
5. Procesor wczyta licznik programu z lokalizacji podanej przez procedurę obsługi przerwań. W zależności od architektury komputera oraz sposobu zaprojektowania systemu operacyjnego, możemy mieć do czynienia z jednym programem typu:
    - jeden program dla każdego rodzaju przerwania,
    - jeden program dla każdego rodzaju urządzenia oraz przerwania.

    Jeżeli dostępnych jest więcej procedur obsługi przerwań, procesor musi podjąć decyzję, którą procedurę wzbudzić.
6. Licznik programu oraz słowo stanu programu odnoszące się do przerwanego programu zostały zapisane na stosie systemowym, ale mogą istnieć inne dane, które są częścią stanu wykonywanego programu. Szczególnie ważne jest zapisanie stanu rejestrów procesora, ponieważ rejestry te mogą być wykorzystywane przez procedurę obsługi przerwań.
7. Program do obsługi przerwania może teraz obsłużyć przerwanie. W tym celu badane są informacje na temat stanu związanego z operacjami we/wy lub innym zdarzeniem.
8. Po obsłużeniu przerwania zapisane wartości rejestru są pobierane ze stosu i przywracane w rejestrach.
9. Na koniec pozostaje przywrócić ze stosu słowo stanu (PSW) oraz licznik programu.

1.5 Hierarchia pamięci

Ograniczenia związane z pamięcią komputera:

- Ile pamięci będzie potrzeba?
  Jeśli tylko istnieje możliwość zwiększenia ilości pamięci, warto to zrobić, gdyż program prędzej czy później będzie jej potrzebował więcej.

- Jaka powinna być jej szybkość?
  Jeśli chodzi o szybkość pamięci, to zawsze pamięć szybsza jest lepsza. Ponieważ system powinien dotrzymywać kroku procesorowi. Jeśli procesor musi czekać na kolejne rozkazy lub argumenty to spada efektywność działania systemu.

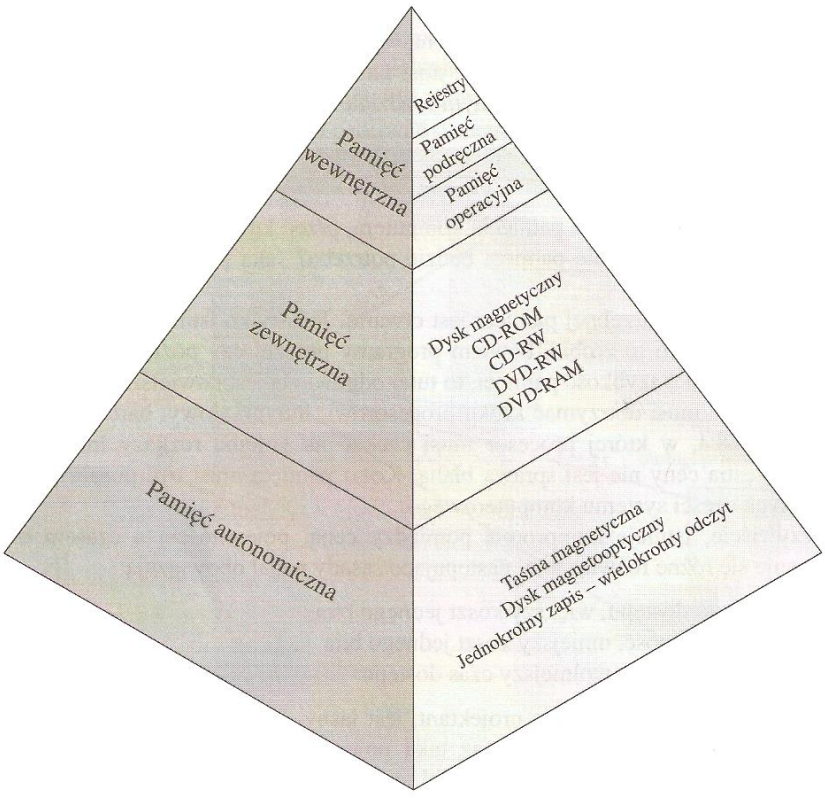
- Ile to będzie kosztować?
  Koszt pamięci musi być proporcjonalny do kosztu innych części systemu komputerowego.

W tworzeniu systemu komputerowego należy starać osiągnąć kompromis pomiędzy pojemnością, czasem dostępu a ceną.

Podstawowe zasady, które obowiązują zawsze w takich rozważaniach to:

- Szybszy czas dostępu, większy koszt jednego bita
- Większa pojemność, mniejszy koszt jednego bita
- Większa pojemność, wolniejszy czas dostępu.

Rozwiązaniem tego problemu jest zrezygnowanie z rozwiązań oferujących tylko jeden składnik pamięci na rzecz **hierarchii pamięci**, którą najczęściej przedstawia się w formie piramidy.



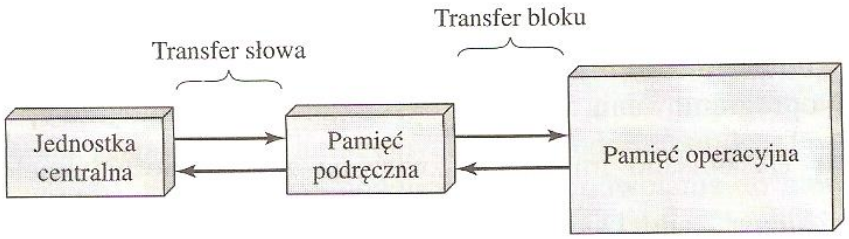
Analizując piramidę z góry w dół, można dojść do następujących wniosków:

- Zmniejsza się koszt jednego bita
- Zwiększa się pojemność
- Zwiększa się czas dostępu
- Zmniejsza się częstotliwość dostępu procesora do pamięci.

Zasada lokalności odniesień (ang. locality of reference) – w trakcie wykonywania programu, odwołania procesora do pamięci, zarówno dla rozkazów jak i danych, mają tendencję do grupowania się. W programach jest to realizowane przez zestawy pętli iteracyjnych oraz podprogramów (funkcji).

1.6 Pamięć podręczna

Pamięć podręczna jest niewidoczna dla systemu operacyjnego, współdziała ona ze sprzętem służącym do zarządzania pamięcią. Wiele zasad wykorzystywanych w schematach pamięci wirtualnej jest również stosowana w pamięci podręcznej.



Motywacja

We wszystkich cyklach rozkazów procesor uzyskuje dostęp do pamięci przynajmniej raz w celu pobrania rozkazu oraz często dodatkowo, aby pobrać argumenty i/lub przechowywać wyniki.

Szybkość wykonywania rozkazów przez procesor jest ograniczona poprzez długość trwania cyklu pamięci (czasu potrzebnego do przeczytania jednego słowa z pamięci lub zapisania do niej). To ograniczenie stanowi poważny problem z uwagi na rozbieżność w szybkości działania procesora i pamięci operacyjnej. (procesor działa znacznie szybciej niż pamięć operacyjna).

Idealnym rozwiązaniem byłoby zbudowanie pamięci operacyjnej w tej samej technologii co rejestry procesora, ale jest to strategia zbyt kosztowna.

Rozwiązaniem tego problemu jest zastosowanie **zasady lokalności odniesień**. Realizuje się to przez stosowanie niewielkiej, szybkiej pamięci umieszczanej pomiędzy procesorem a pamięcią operacyjną.

Taką pamięć nazywamy **pamięcią podręczną.**

Zasada działania pamięci podręcznej

Szybkość działania pamięci podręcznych jest porównywalna z szybkością najszybszych pamięci przy jednoczesnym zachowaniu dużych pojemności.

Pamięć podręczna zawiera kopię części pamięci operacyjnej. Kiedy procesor próbuje odczytać bajt z pamięci, najpierw sprawdza, czy bajt nie znajduje się w pamięci podręcznej. Jeśli tak zostaje on natychmiast dostarczony procesorowi. W przeciwnym wypadku część pamięci operacyjnej (składająca się ze stałej liczby bajtów) jest przenoszona do pamięci podręcznej i następnie dostarczona do procesora.

Z zasady lokalności odniesień wynika, że jeśli do pamięci podręcznej zostanie wstawiony blok danych z uwagi na odwołanie do jednego bajta w tym bloku, jest bardzo prawdopodobne, że wiele kolejnych odniesień również odwoła się do bajtów zawartych w tym bloku.

Projektowanie pamięci podręcznej

Projektowanie pamięci podręcznej dotyczy podobnych kwestii co projektowanie pamięci wirtualnej.

Kwestie te można podzielić na następujące kategorie:

- Rozmiar pamięci podręcznej
  Okazuje się, że względnie niewielkie pamięci podręczne w znacznym stopniu poprawiają wydajność systemu komputerowego.
- Rozmiar bloku – jednostka danych przenoszona pomiędzy pamięcią podręczną a pamięcią operacyjną.
- Funkcja odwzorowania – określa docelową lokalizację pamięci podręcznej dla bloku danych odczytanych i przenoszonych z pamięci operacyjnej.

- Algorytm zastępowania – podejmuje decyzję (przy narzuconych mu ograniczeniach), który blok danych w pamięci podręcznej musi zostać wymieniony z uwagi na konieczność wstawienia nowego bloku z pamięci operacyjnej.
- Polityka zapisu – determinuje sytuację, w której ma miejsce operacja zapisu do pamięci podręcznej po pobraniu nowych danych z pamięci operacyjnej.

1.7 Mechanizmy komunikacji na we/wy

Istnieją trzy mechanizmy realizacji operacji we/wy:

1. Programowe we/wy

Kiedy procesor wykonuje program lub napotka rozkaz odnoszący się do operacji we/wy, realizuje taki rozkaz poprzez wysłanie polecenia do odpowiedniego modułu we/wy.

W przypadku programowego we/wy, moduł we/wy wykonuje żądaną czynność i ustawia odpowiednie bity w rejestrze stanu modułu we/wy. Jednak żaden alert nie dociera do procesora, a działanie procesora nie ulega przerwaniu. Po ujawnieniu się rozkazu we/wy procesor musi podjąć aktywne działanie, by określić, kiedy ten rozkaz może być wykonany. W tym celu procesor od czasu do czasu sprawdza status modułu we/wy, aż uzyska informację, że dana operacja we/wy została zakończona.

Kategorie zestawu rozkazów we/wy:

- **Sterowanie** – służy do aktywowania zewnętrznego urządzenia i sterowania nim.
- **Status** – służy do testowania różnorodnych rodzajów statusu skojarzonych z modułem we/wy oraz jego urządzeniami peryferyjnymi.
- **Transfer** – wykorzystywany do odczytu i/lub zapisu danych pomiędzy procesorem a rejestrami oraz urządzeniami zewnętrznymi.

2. Układ we/wy sterowany przerwaniami

Problem związany z mechanizmem programowego we/wy polega na tym, że procesor musi długo czekać, aż potrzeby moduł we/wy będzie gotów na przyjęcie lub transmisję większej części danych. Podczas oczekiwania na zgłoszenie gotowości modułu we/wy procesor musi nieustannie sprawdzać jego status. W rezultacie wydajność całego systemu znacząco spada.

Alternatywnie procesor może wysłać do modułu polecenie we/wy i zająć się obsługą innych zadań. Moduł we/wy przerwie pracę procesora, by zażądać usługi, gdy będzie gotów na wymianę danych z procesorem. Procesor wykonuje rozkaz transferu danych, po czym wraca do poprzednio obsługiwanego zadania.

W komputerach znajduje się wiele modułów we/wy, dlatego potrzebne są mechanizmy ułatwiające procesorowi stwierdzenie, które urządzenie wystosowało przerwanie. W przypadku pojawienia się jednocześnie wielu przerwań procesor musi dodatkowo określić kolejność ich wykonania.

3. Bezpośredni dostęp do pamięci (DMA)

Układy we/wy sterowane przerwaniami są wydajniejsze od programowych układów we/wy, to jednak nadal wymagane jest pośrednictwo procesora w celu przeniesienia danych pomiędzy pamięcią a modułem we/wy. Dane muszą być kopiowane z wykorzystaniem procesora.

Okazuje się, że obie formy komunikacji (programowa i sprzętowa) z wykorzystaniem układów we/wy nieodłącznie wiążą się z dwoma minusami:

- Prędkość transferu we/wy jest ograniczona prędkością, z jaką procesor może testować i obsługiwać dane urządzenie.
- Procesor jest zajęty obsługą transferu we/wy. Każdy transfer danych wiąże się z wystosowaniem wielu rozkazów.

DMA (ang. Direct Memory Access) – oddzielny moduł lub część modułu we/wy zapewniający bezpośredni dostęp do pamięci. Wykorzystywany w razie potrzeby przekopiowania dużej porcji danych.

W każdym przypadku działa następująco:

Kiedy procesor chce odczytać lub zapisać blok danych, wysyła polecenie do modułu DMA z następującymi informacjami:

- rodzajem żądanego polecenia: odczyt/zapis
- adresem urządzenia we/wy
- początkową lokalizacją w pamięci, gdzie zostanie przeprowadzona operacja zapisu/odczytu
- liczba słów, które mają zostać odczytane/zapisane.

Moduł DMA musi przejąć sterowanie magistralą systemową, przetransferować dane z i do pamięci. Z uwagi na to, że wiele urządzeń odwołuje się do magistrali, może się zdarzyć, że procesor potrzebuje magistrali do realizacji innych zadań i musi poczekać na zgłoszenie modułu DMA. Takie działanie nie jest przerwaniem – bo procesor nie zapisuje kontekstu i nie obsługuje innych zadań. Zatrzymanie procesora trwa jeden cykl magistrali.

W przypadku transferu wielu słów mechanizm DMA jest o wiele wydajniejszy od układu we/wy sterowanego przerwaniami lub programowego we/wy.

Pytania podsumowujące ten rozdział:

1. Wymień i pokrótce omów cztery główne elementy komputera.
2. Zdefiniuj dwie główne kategorie rejestrów procesora.
3. Co to jest przerwanie?
4. W jaki sposób rozwiązano problem wielu przerwań?
5. Jakie są cechy charakterystyczne różnych elementów hierarchii pamięci?
6. Co to jest pamięć podręczna?
7. Wymień i krótko opisz trzy mechanizmy operacji na urządzeniach we/wy.