# Systemy operacyjne 1

## Wspomaganie sprzętowe

Arkadiusz Chrobot

Katedra Systemów Informatycznych, Politechnika Świętokrzyska w Kielcach

Kielce, 11 października 2024

# Plan wykładu

# Plan wykładu

# Plan wykładu

# Plan wykładu

# Plan wykładu

# Plan wykładu

# Plan wykładu

# Plan wykładu

1. Scenariusze obsługi wejścia-wyjścia
2. System przerwań
   - Sposób działania
   - Realizacja
   - DMA
   - Wyjątki
   - Zagadnienia pokrewne
3. Ochrona sprzętowa
   - Dualny tryb pracy procesora
   - Ochrona pamięci
   - Timer
4. Wywołania systemowe
5. Architektury wieloprocesorowe
6. Komputery osobiste

# Plan wykładu

1. Scenariusze obsługi wejścia-wyjścia
2. System przerwań
   - Sposób działania
   - Realizacja
   - DMA
   - Wyjątki
   - Zagadnienia pokrewne
3. Ochrona sprzętowa
   - Dualny tryb pracy procesora
   - Ochrona pamięci
   - Timer
4. Wywołania systemowe
5. Architektury wieloprocesorowe
6. Komputery osobiste

# Plan wykładu

1. Scenariusze obsługi wejścia-wyjścia
2. System przerwań
   - Sposób działania
   - Realizacja
   - DMA
   - Wyjątki
   - Zagadnienia pokrewne
3. Ochrona sprzętowa
   - Dualny tryb pracy procesora
   - Ochrona pamięci
   - Timer
4. Wywołania systemowe
5. Architektury wieloprocesorowe
6. Komputery osobiste

# Plan wykładu

1. Scenariusze obsługi wejścia-wyjścia
2. System przerwań
   - Sposób działania
   - Realizacja
   - DMA
   - Wyjątki
   - Zagadnienia pokrewne
3. Ochrona sprzętowa
   - Dualny tryb pracy procesora
   - Ochrona pamięci
   - Timer
4. Wywołania systemowe
5. Architektury wieloprocesorowe
6. Komputery osobiste

# Plan wykładu

1. Scenariusze obsługi wejścia-wyjścia
2. System przerwań
   - Sposób działania
   - Realizacja
   - DMA
   - Wyjątki
   - Zagadnienia pokrewne
3. Ochrona sprzętowa
   - Dualny tryb pracy procesora
   - Ochrona pamięci
   - Timer
4. Wywołania systemowe
5. Architektury wieloprocesorowe
6. Komputery osobiste

# Plan wykładu

1. Scenariusze obsługi wejścia-wyjścia
2. System przerwań
   - Sposób działania
   - Realizacja
   - DMA
   - Wyjątki
   - Zagadnienia pokrewne
3. Ochrona sprzętowa
   - Dualny tryb pracy procesora
   - Ochrona pamięci
   - Timer
4. Wywołania systemowe
5. Architektury wieloprocesorowe
6. Komputery osobiste

# Plan wykładu

1. Scenariusze obsługi wejścia-wyjścia
2. System przerwań
   - Sposób działania
   - Realizacja
   - DMA
   - Wyjątki
   - Zagadnienia pokrewne
3. Ochrona sprzętowa
   - Dualny tryb pracy procesora
   - Ochrona pamięci
   - Timer
4. Wywołania systemowe
5. Architektury wieloprocesorowe
6. Komputery osobiste

# Obsługa wejścia-wyjścia

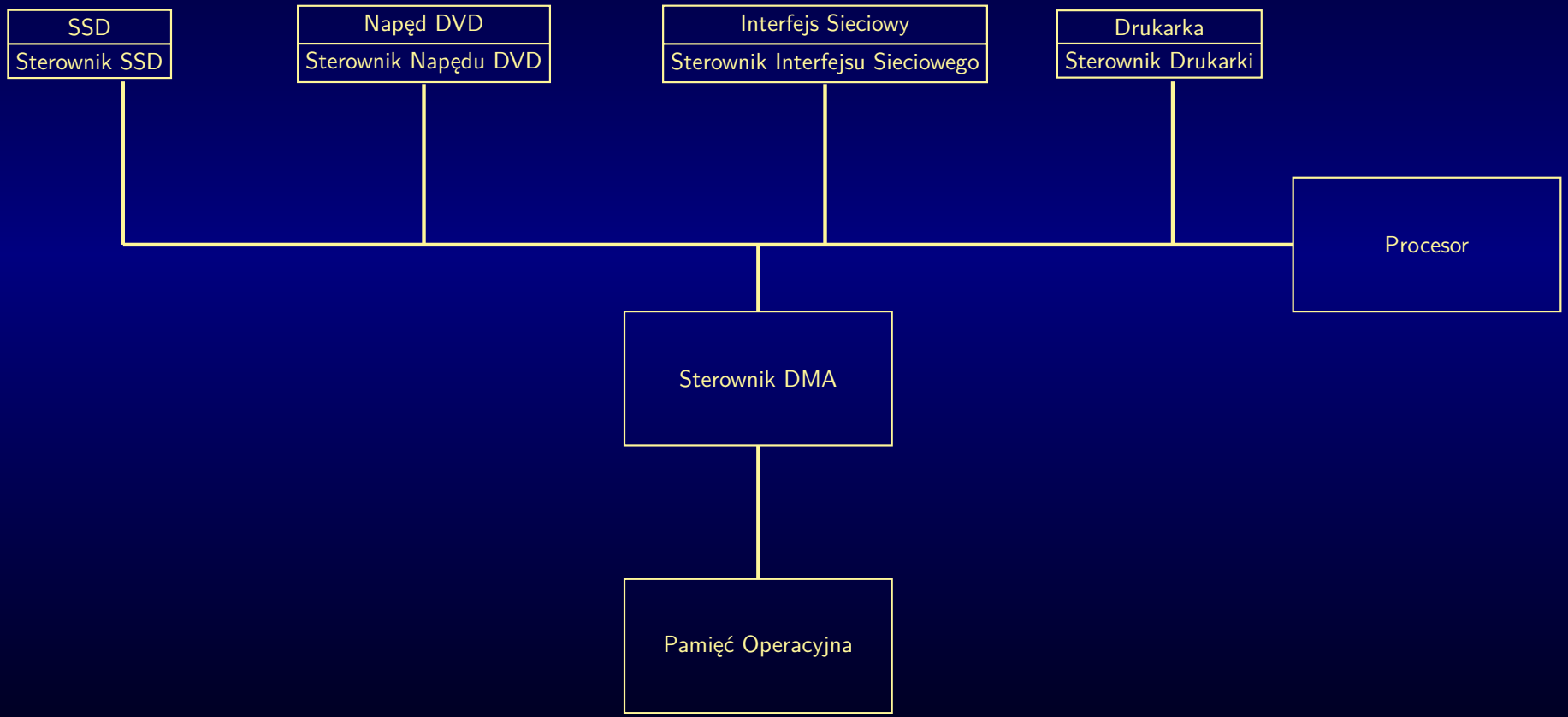
## Aktywne oczekiwanie (ang. *busy waiting*)

Procesor nadzoruje pracę urządzenia przez cały czas trwania transmisji. Nadzór ten obejmuje takie czynności, jak: sprawdzenie gotowości urządzenia, wprowadzenie danych niezbędnych do wykonania komunikacji, **oczekiwanie na odbiór lub wysłanie danych**, w przypadku odbioru skopiowanie danych do pamięci operacyjnej. **W czasie oczekiwania na zakończenie operacji wejścia-wyjścia procesor nie wykonuje żadnych innych zadań.**

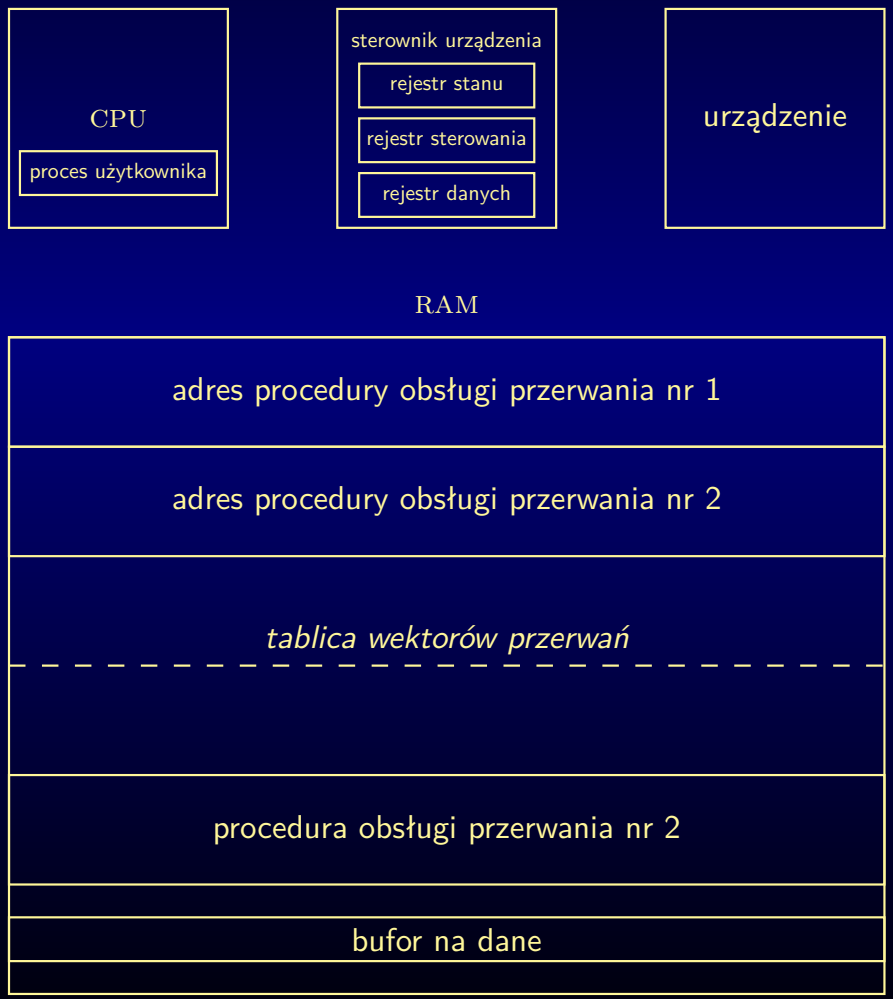
## Przerwania (ang. *interrupts*)

Urządzenie możemy wyposażyć w układ sterownika, który może nadzorować jego pracę, zwalniając tym samym procesor z tego obowiązku. **W szczególności CPU nie musi czekać na zakończenie transmisji.** Sterownik powiadamia procesor o tym zdarzeniu generując sygnał nazywany *przerwaniem*.

# Przerwania — schemat sprzętu

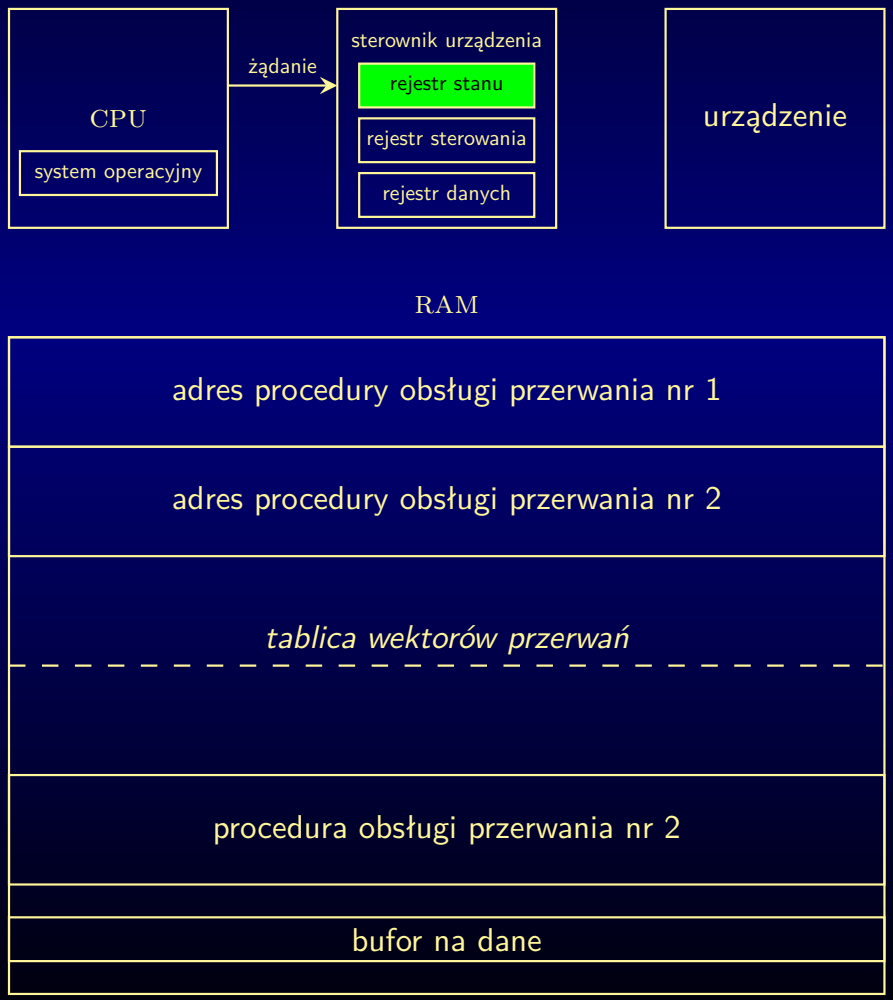


# Przerwania — schemat działania



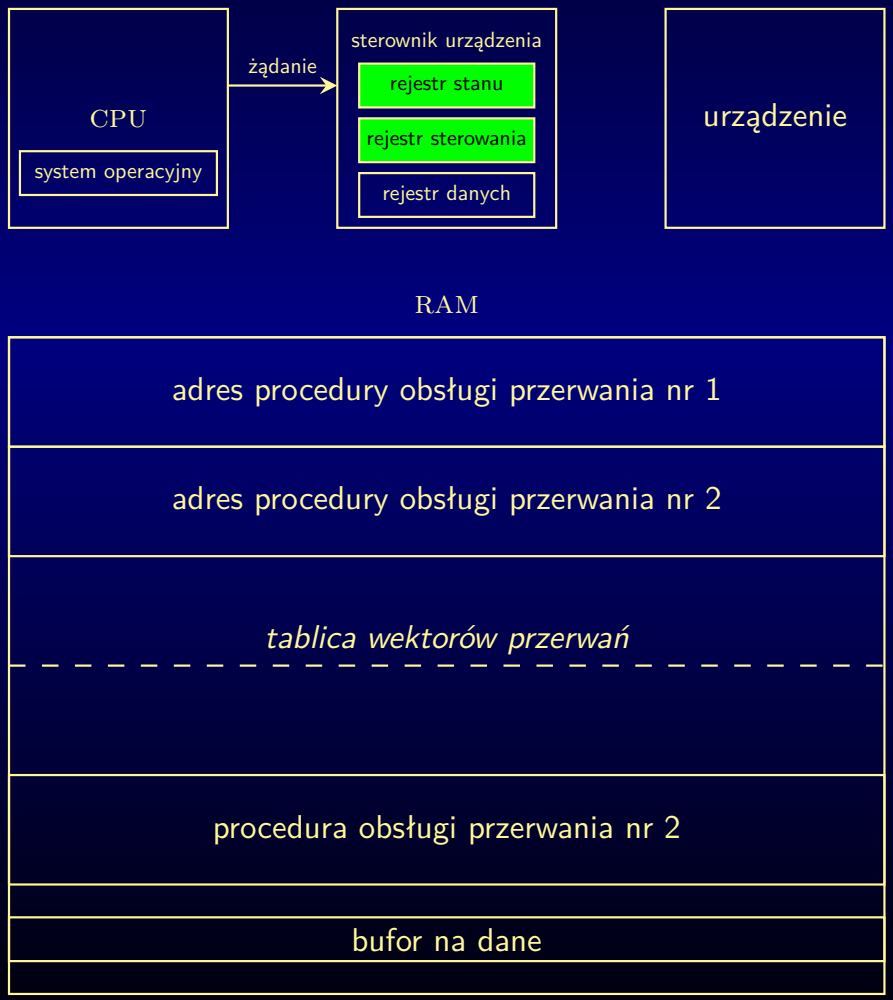

Proces (program) użytkownika żąda od systemu operacyjnego wykonania komunikacji z urządzeniem peryferyjnym, polegającej na przesłaniu danej z urządzenia.

# Przerwania — schemat działania



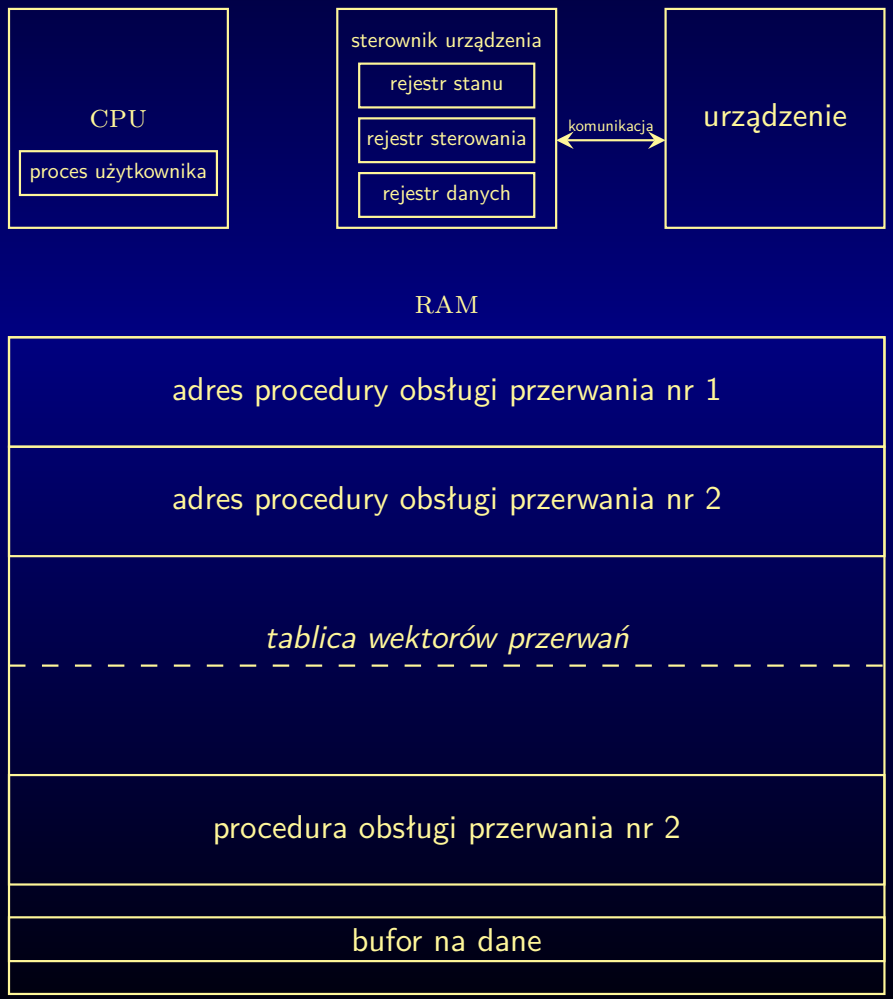

System operacyjny sprawdza, czy urządzenie jest gotowe do transmisji badając zawartość rejestru stanu sterownika urządzenia.

# Przerwania — schemat działania



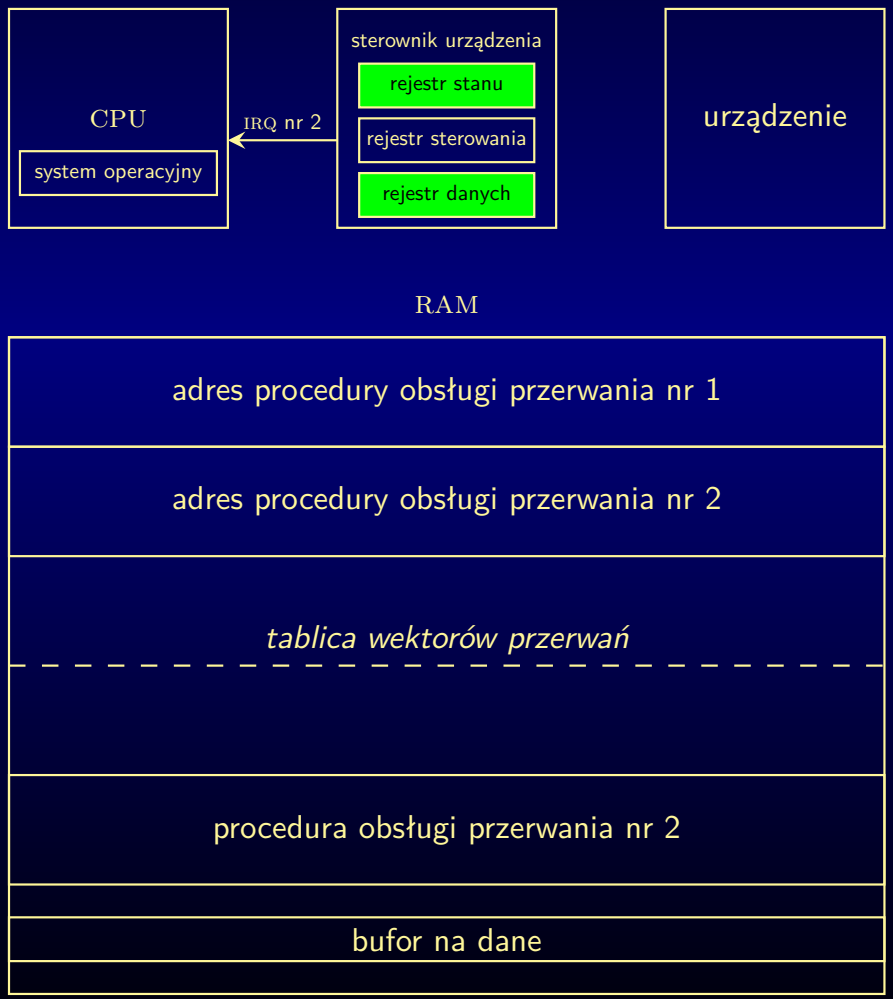

System operacyjny programuje sterownik urządzenia wpisując odpowiedni rozkaz do jego rejestru sterowania.

# Przerwania — schemat działania



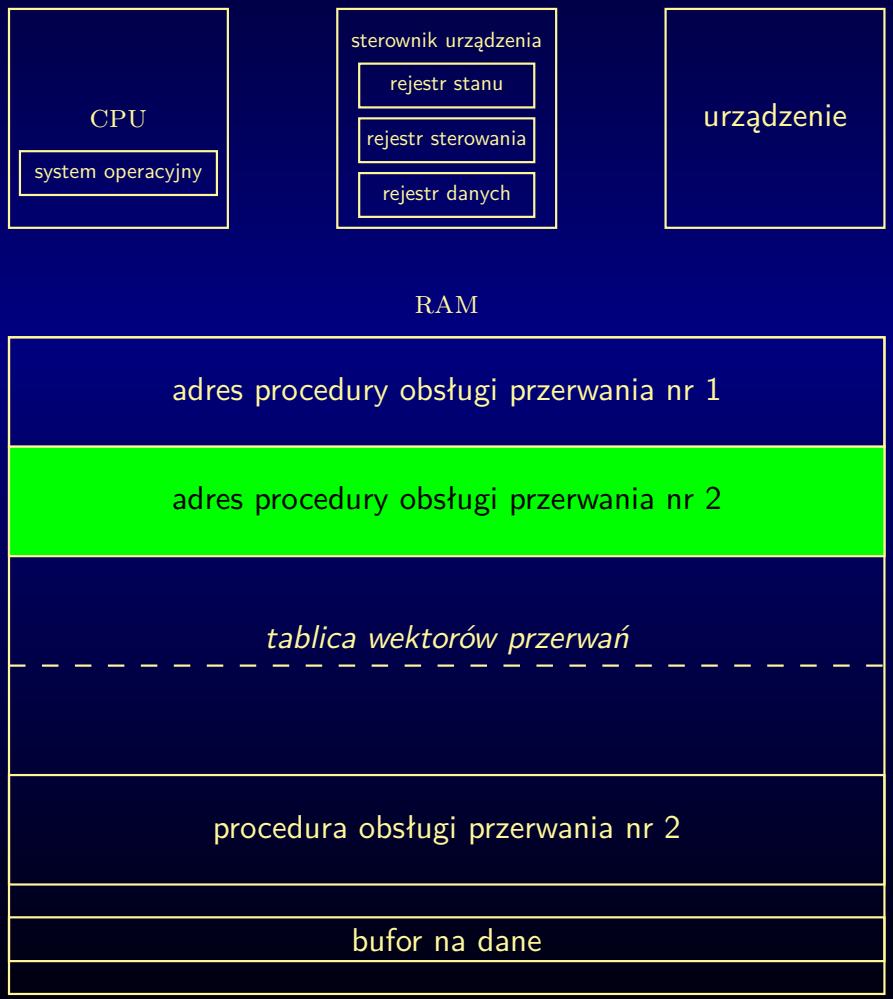

System operacyjny oddaje procesor procesowi użytkownika. W zależności od rodzaju wykonywanej operacji wejścia-wyjścia może to być ten sam proces, który zażądał wykonania transmisji, lub inny. W tym samym czasie sterownik nadzoruje pracę urządzenia, bez interwencji procesora.

# Przerwania — schemat działania



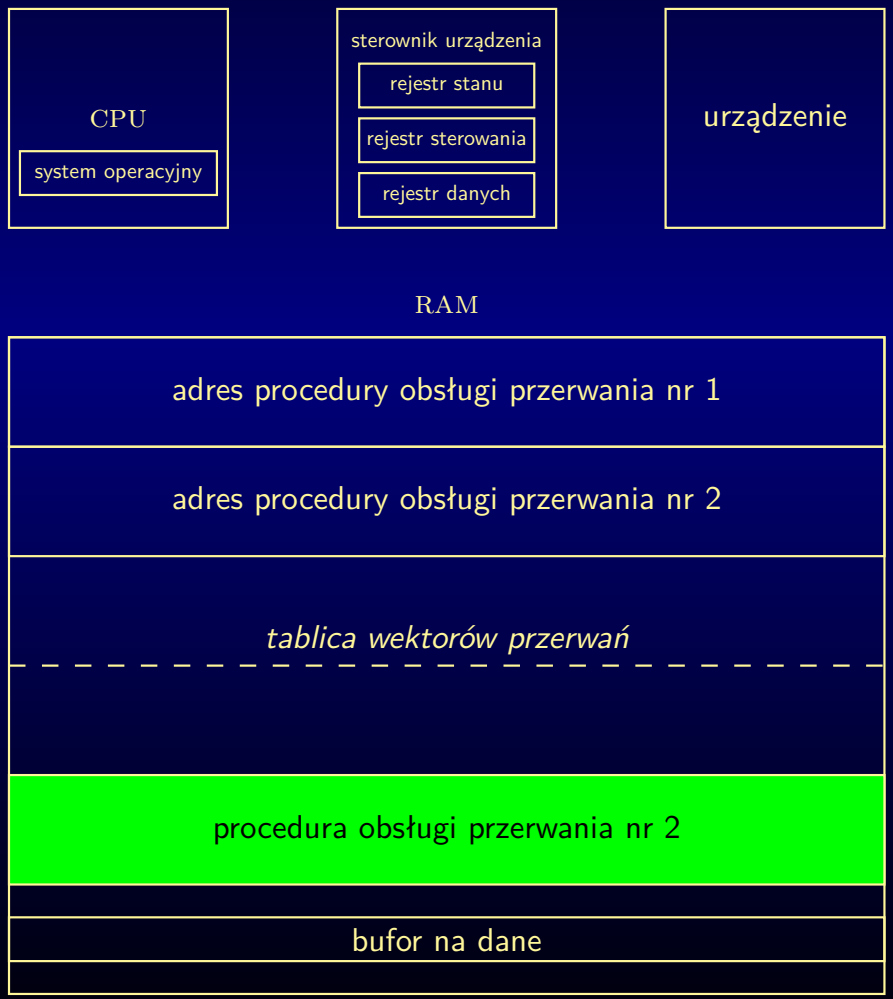

Kończy się proces transmisji, odebrana informacja jest umieszczona w rejestrze danych sterownika. W rejestrze stanu sterownik ustawia flagę oznaczającą zakończenie komunikacji, a następnie zgłasza przerwanie. Powoduje to automatyczne zapamiętanie bieżącego stanu procesora i adresu powrotu oraz przekazanie sterowania do systemu operacyjnego (objaśnione na kolejnych slajdach).

# Przerwania — schemat działania



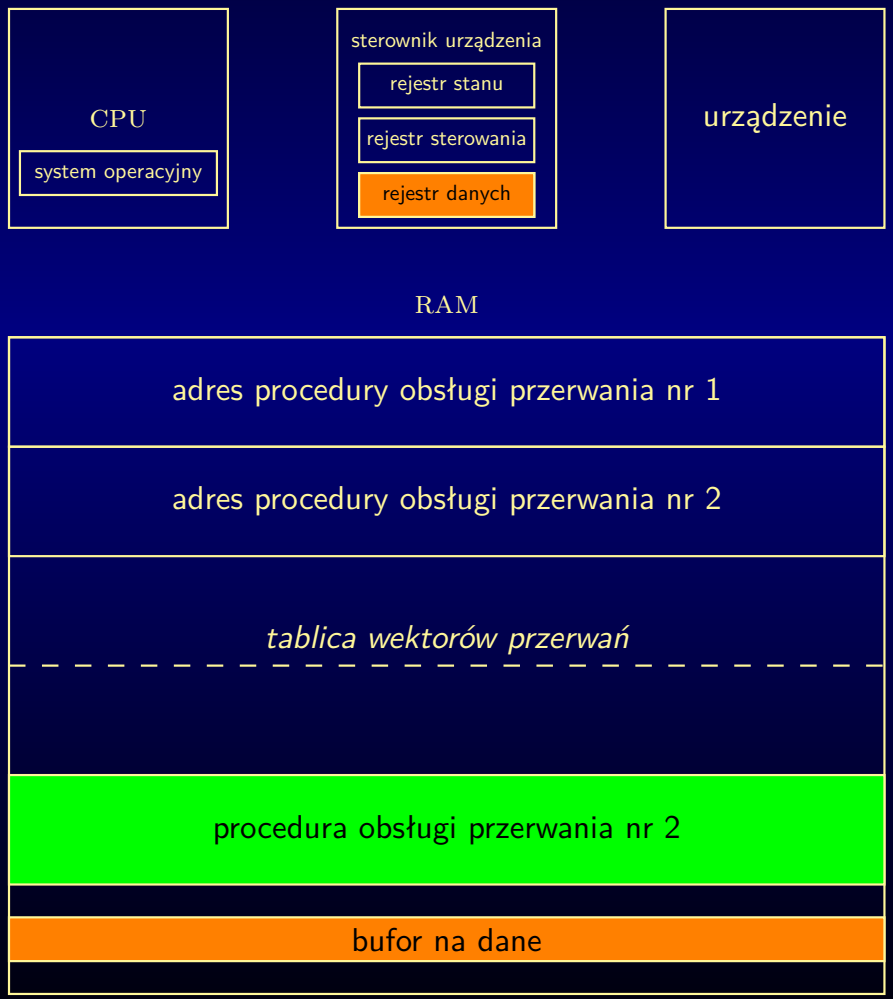

Następuje identyfikacja źródła przerwania — w tym przypadku dzięki wektorowemu systemowi przerwań. Z tablicy wektorów przerwań pobierany jest adres procedury obsługi tego przerwania.

# Przerwania — schemat działania



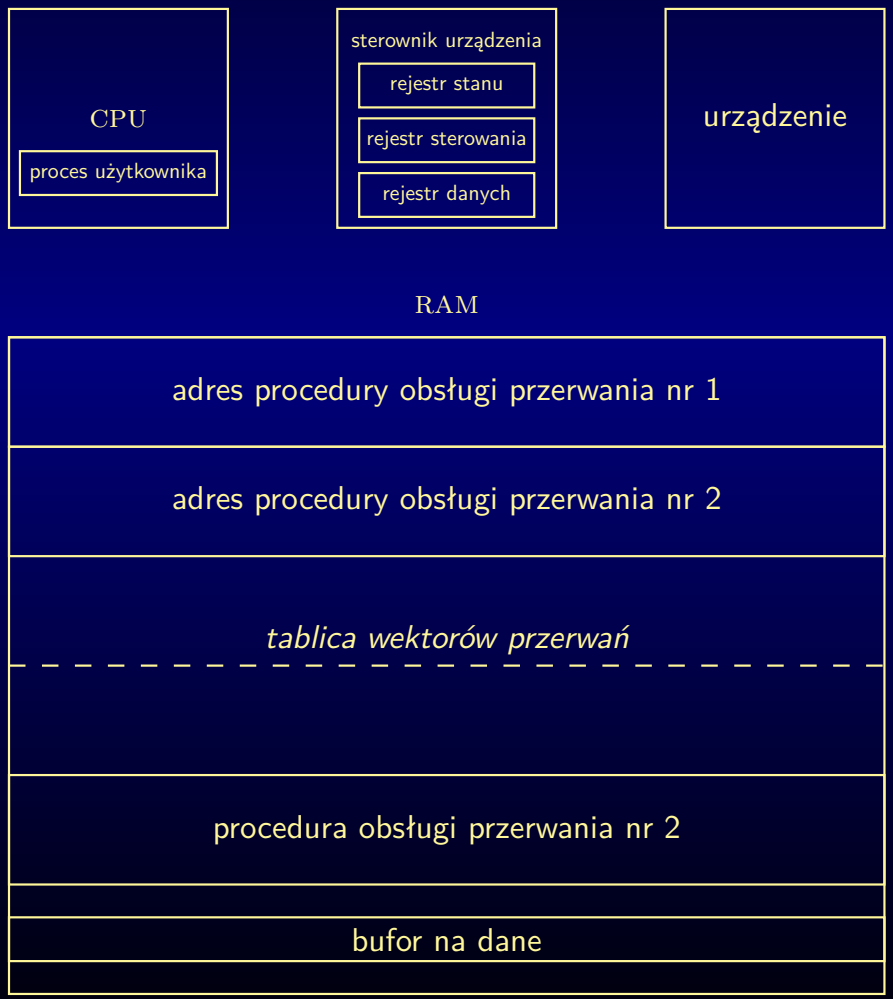

Sterowanie zostaje przekazane do procedury obsługi przerwania.

# Przerwania — schemat działania




Podprogram obsługi przerwania kopiuje zawarość rejestru danych sterownika do bufora w pamięci i kończy swoją pracę.

# Przerwania — schemat działania




Procesor wraca do przerwanego zadania.

# System przerwań — implementacja.

Podstawowym problemem, po wystąpieniu przerwania, jest zidentyfikowanie jego źródła. Pierwsze procesory, w których zastosowano przerwania miały tylko jedną linię zgłoszenia przerwania (*IRQ*), do której podłączone były wszystkie urządzenia peryferyjne. Jeśli wystąpił sygnał przerwania, to procesor sprawdzał rejestry stanu wszystkich urządzeń, celem znalezienia tego, które zgłosiło żądanie obsługi i uruchamiał odpowiadającą mu procedurę obsługi. Ten tryb ustalania źródła przerwania nazywamy odpytywaniem (ang. *polling*). Procesor musiał również rozstrzygnąć sytuację, w której więcej niż jedno urządzenie zgłosiło konieczność obsługi.

# System przerwań — implementacja.

Bardziej wydajne rozwiązanie polega na zwiększeniu liczby linii zgłaszania przerwań. Każda z nich otrzymuje unikatowy numer i priorytet. Dzięki temu możliwe jest nie tylko szybkie znalezienie źródła przerwania, ale również określenie kolejności obsługi zgłoszeń przerwań. Zgłoszenie określonego przerwania blokuje możliwość zgłoszenia przerwań o niższym, ale możliwe jest zgłaszanie przerwań o wyższym priorytecie. Wadą tego rozwiązania jest to, że priorytety przerwań są przypisane urządzeniom „na sztywno”.

# System przerwań — implementacja.

Zmianę schematu priorytetów umożliwia zastosowanie programowalnego kontrolera przerwań (ang. PIC). Ten kontroler połączony jest z procesorem. Jeśli urządzenie peryferyjne zgłosi przerwanie, to kontroler je identyfikuje i przekazuje jego numer procesorowi. Numer ten, tak jak w rozwiązaniu opisanym na poprzednim slajdzie, jest traktowany jako indeks w *tablicy wektorów przerwań* (ang. *Interrupt Vector Table* — IVT). Wartościami elementów tej tablicy są adresy procedur obsługi przerwań. Każde przerwanie może więc mieć własną procedurę obsługi. Tablica ta tradycyjnie była umieszczona na początku lub końcu pamięci operacyjnej. Nowsze komputery pozwalają systemowi operacyjnemu zdecydować o jej położeniu. Tablica wektorów przerwań wraz z procedurami obsługi przerwań stanowi część systemu operacyjnego.

# System przerwań — implementacja.

Nowsze platformy PC stosują rozwiązanie będące połączeniem odpytywania i wektorowego systemu przerwań. Linie żądania przerwań, które są przypisane przerwaniom o niskim priorytecie mogą być współdzielone przez kilka urządzeń równocześnie. Przez odpytywanie ustala się, które urządzenie zgłosiło przerwanie. Przerwania o wysokim priorytecie zazwyczaj nie są współdzielone.

# System przerwań — implementacja.

Innym problemem jest zapamiętanie kontekstu procesora, aby mógł on wrócić do zadania które realizował przed wystąpieniem przerwania. Przez kontekst procesora rozumiemy adres w pamięci operacyjnej (*adres powrotu*) określający miejsce, z którego ma być pobrany następny, po zakończeniu procedury obsługi przerwania, rozkaz do realizacji oraz stan rejestrów procesora. Najmniejszy kontekst obejmuje adres powrotu i rejestr stanu maszyny[1]. Wczesne rozwiązania polegały na wyznaczeniu pewnego niewielkiego, ustalonego miejsca w pamięci komputera, gdzie kontekst był składowany. Nowsze polegają najczęściej na zapamiętaniu kontekstu na stosie.

[1] W przypadku procesorów bazujących na architekturze x86 jest to rejestr flag.

# System przerwań — implementacja.

W zależności od tego w jaki sposób zapamiętywany jest kontekst procesora i jak rozwiązana jest kwestia identyfikacji źródła przerwania, możemy różnie obsługiwać sytuację, w której przerwania następują szybko po sobie. Najprostsze rozwiązanie polega na wyłączeniu systemu przerwań na czas realizacji procedury obsługi pierwszego zgłoszonego przerwania i ponownym jego włączeniu po jej zakończeniu. Jest to jedyny sposób, który możemy stosować w systemach z odpytywaniem, ale jego zastosowanie nie ogranicza się wyłącznie do nich. W systemach ze współdzieleniem przerwań może być wyłączana tylko linia współdzielona. Systemy priorytetowe pozwalają na selektywne wyłączanie przerwań o niższym lub równym priorytecie. Technika ta nazywa się *maskowaniem*. Możliwe jest również odkładanie na później ich realizacji.

# System przerwań — implementacja.

We współczesnych systemach operacyjnych, celem zminimalizowania czasu, kiedy określona grupa przerwań lub wszystkie przerwania są wyłączone, procedury obsługi przerwań są podzielone na dwie części zwane połówkami. Górna połówka jest procedurą wykonywaną zaraz po otrzymaniu przerwania. Jej działanie jest krótkie i najczęściej sprowadza się do potwierdzenia odebrania przerwania oraz inicjacji dolnej połówki, której uruchomienie może być odroczone i która wykonuje czasochłonne czynności obsługi przerwania. Więcej o różnych rozwiązaniach dotyczących obsługi przerwań można dowiedzieć się z artykułu Marka Smothermana pt. ▶ *Interrupts*.

# Direct Memory Access (DMA)

Opisany wcześniej sposób komunikacji z urządzeniami wejścia-wyjścia, sprawdza się w przypadku dosyć wolnych jednostek, jak np. klawiatura. Takie urządzenia nazywane są *urządzeniami znakowymi*. Gdyby zastosować go do szybkich urządzeń, takich jak np. dysk twardy to okazałoby się, że procesor musiałby większość czasu spędzać wykonując podprogramy obsługi przerwań, ze względu na bardzo dużą liczbę ich zgłoszeń. Rozwiązaniem jest zastosowanie dla tych urządzeń bezpośredniego dostępu do pamięci (*DMA*). System komputerowy jest *wyposażany* w dodatkowy sterownik zwany *kontrolerem DMA*. Kiedy program użytkownika żąda transmisji danych z szybkiego urządzenia to system operacyjny programuje odpowiednio kontroler DMA, co wymaga między innymi wyznaczenia obszaru w pamięci do którego urządzenie będzie zapisywało dane (lub z którego będzie pobierało je w przypadku transmisji w odwrotnym kierunku) oraz określenia wielkości tych danych. Urządzenie przesyła te dane blokami wielkości kilkuset lub nawet kilku tysięcy bajtów. Sterownik DMA powiadamia odpowiednim przerwaniem procesor, że transmisja została zakończona. Procesor nie nadzoruje przesyłania danych do lub z pamięci, robi to kontroler DMA. Urządzenia obsługiwane w ten sposób są nazywane *urządzeniami blokowymi*.

# Wyjątki

Wektorowy system przerwań można wykorzystać nie tylko do obsługi komunikacji z urządzeniami zewnętrznymi, ale również do obsługi sytuacji wyjątkowych. Najprostszym przykładem takiej sytuacji jest próba dzielenia przez zero. Przerwania obsługujące zdarzenia tego typu nazywamy pułapkami (ang. *trap*) lub wyjątkami (ang. *exception*). Takie przerwania mają wysokie priorytety i są najczęściej *niemaskowalne*. Pułapki są najczęściej przerwaniami *synchronicznymi*, tzn. mogą pojawić się podczas realizacji określonych fragmentów programu, podczas gdy przerwania związane z urządzeniami peryferyjnymi mogą pojawić się podczas realizacji dowolnej instrukcji programu. Dlatego te ostatnie nazywane są przerwaniami *asynchronicznymi*. Od reakcji na sytuacje wyjątkowe zależy stabilność pracy systemu komputerowego.

# Obsługa wielu żądań dostępu do urządzenia

Ponieważ działanie systemu operacyjnego w dużej mierze zależy od systemu przerwań, łatwo zauważyć, że jest on *oprogramowaniem reagującym na zdarzenia*. Projektując system operacyjny należy uwzględnić sytuację, w której urządzenie wejścia-wyjścia nie jest w stanie obsłużyć szybko otrzymywanych od procesora zleceń. Jest to sytuacja do której dochodzi bardzo często. Ten problem rozwiązuje się tworząc kolejki zleceń przypisane dla każdego urządzenia w komputerze. Jeżeli urządzenie nie jest zajęte realizacją innych zleceń, to od razu przystępuje do realizacji skierowanego do niego zlecenia, w przeciwnym przypadku zlecenie to trafia do kolejki, gdzie oczekuje na swoją realizację.

# Rozkaz oczekiwania

Pozostaje do rozstrzygnięcia kwestia, co powinien zrobić system operacyjny z procesem, którego żądanie operacji wejścia-wyjścia oczekuje na realizację. Jeśli żądane dotyczyło zapisu lub wysłania danych, to może on kontynuować swoją pracę. W przypadku operacji odbioru bądź odczytu danych system operacyjny może mu pozwolić na aktywne oczekiwanie (co jest jawnym marnotrawieniem czasu procesora) lub umieścić go w kolejce oczekiwania na zakończenie zleconej operacji wejścia-wyjścia, a procesor oddać innemu procesowi. W przypadku *systemów wbudowanych* (ang. *embedded systems*) dosyć często zdarza się, że wszystkie procesy oczekują na zakończenie operacji wejścia-wyjścia lub innego *zdarzenia* sygnalizowanego przerwaniem. Dlatego procesory w tego typu systemach wyposażone są w rozkaz, nazywany najczęściej `wait`, który zawiesza ich pracę do czasu pojawienia się przerwania.

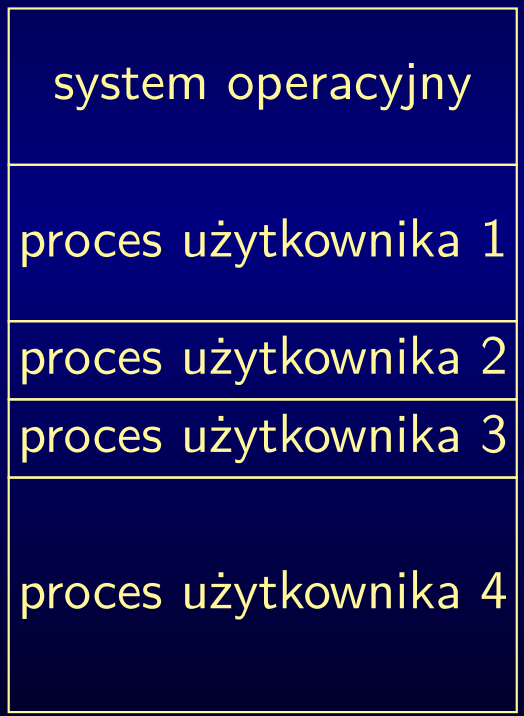
# Prosta ochrona sprzętowa

Zastosowanie pułapek nie gwarantuje samoistnie stabilności systemu. Muszą istnieć dodatkowe mechanizmy, które pozwolą na jej zachowanie. Trzy z nich są konieczne do zbudowania najprostszego systemu ochrony.

# Dualny tryb pracy procesora

Procesy użytkownika nie powinny mieć możliwości wykonywania pewnych rozkazów. Aby to zapewnić twórcy sprzętu wyposażyli procesory w dwa tryby pracy: tryb jądra (nazywany też trybem monitora, systemu lub nadzorcy) i tryb użytkownika. W pierwszym trybie wykonywany jest system operacyjny, w drugim, procesy użytkownika. Przełączenie z trybu użytkownika do trybu monitora następuje w wyniku wystąpienia przerwania lub pułapki. To w jakim trybie znajduje się w chwili obecnej procesor określone jest ustawieniem odpowiedniego bitu w rejestrze stanu procesora. Lista rozkazów jest podzielona na dwa zbiory: rozkazów uprzywilejowanych, które mogą być wykonywane jedynie w trybie jądra i nieuprzywilejowanych, które mogą być wykonywane w obu trybach. Rozróżnia je ustawienie odpowiedniego bitu w słowie rozkazu, który jest porównywany z bitem trybu pracy. Jeśli program użytkownika spróbuje wykonać rozkaz uprzywilejowany, to zostanie uruchomiona odpowiednia pułapka, która awaryjnie kończy jego działanie.

# Prosta ochrona pamięci




W systemach wielozadaniowych system operacyjny i pewna liczba programów użytkownika stale rezyduje w pamięci operacyjnej komputera.

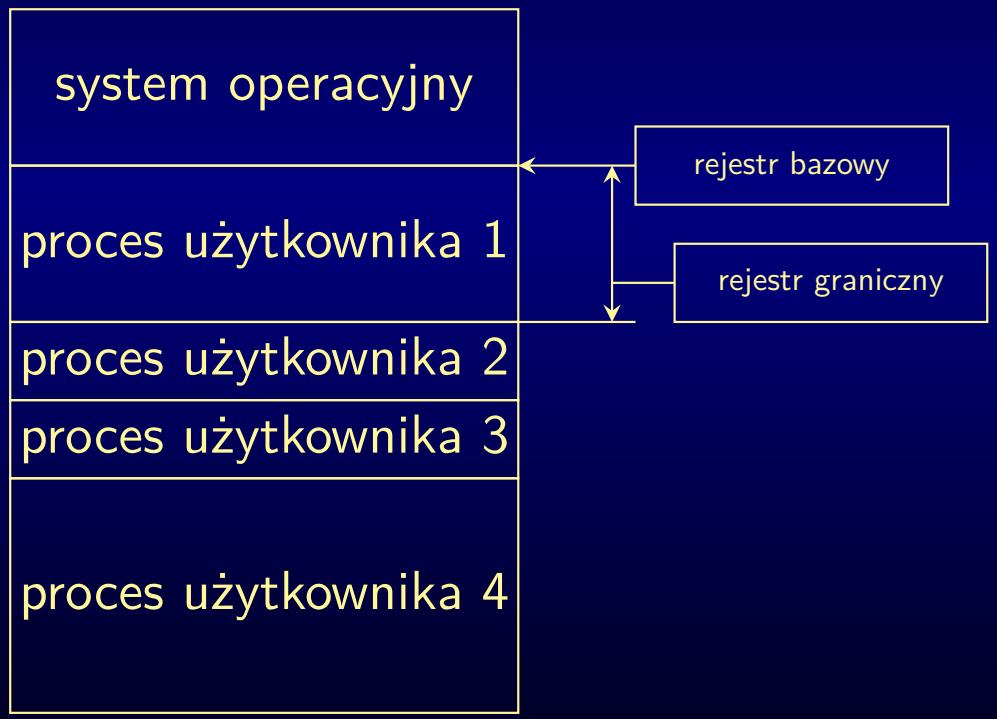
# Prosta ochrona pamięci


Jeśli wśród tych programów znalazłby się program, który odwoływałby się do pamięci w nieprawidłowy sposób, bądź to na skutek błędu programisty, bądź na skutek celowego, złośliwego działania...

# Prosta ochrona pamięci


...wówczas cały system mógłby być zagrożony.

# Prosta ochrona pamięci

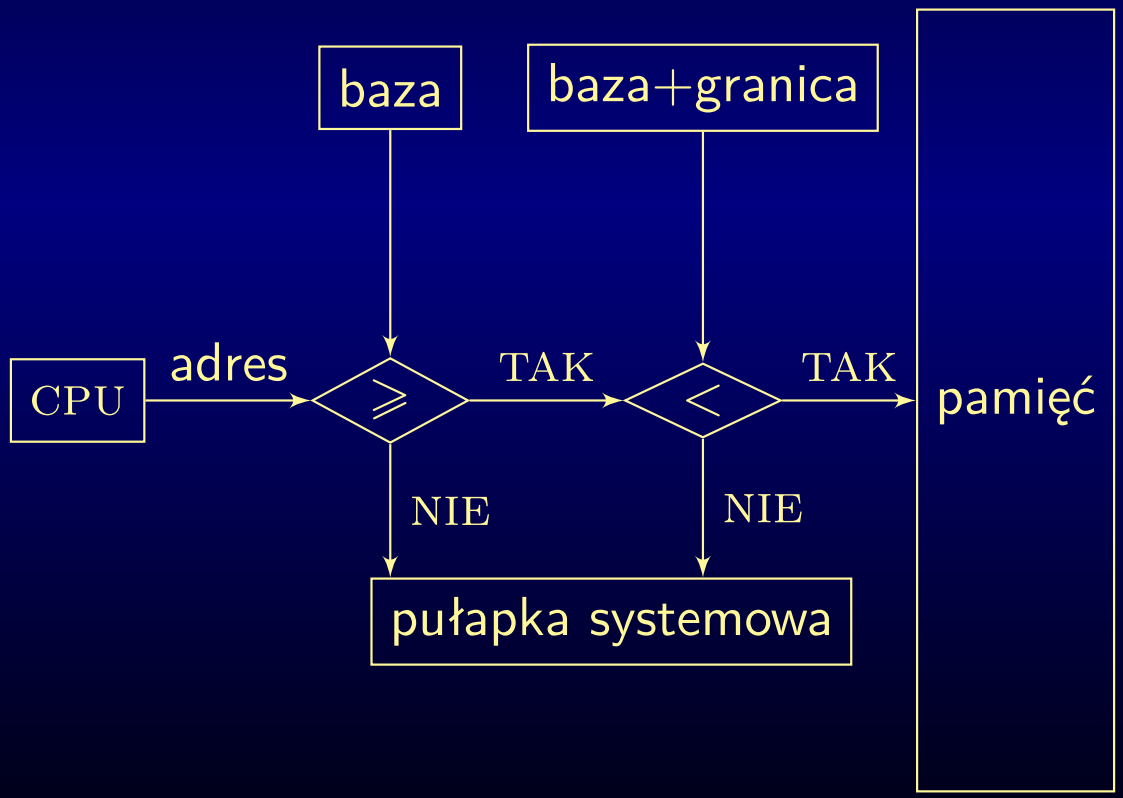



Aby zapobiec takiej sytuacji należy kontrolować każde odwołanie do pamięci bieżąco wykonywanego procesu użytkownika. Najprostsze rozwiązanie polega na zastosowaniu odpowiedniego układu elektronicznego składającego się między innymi z dwóch rejestrów — bazowego i granicznego. Pierwszy zawiera adres początku obszaru pamięci, który system operacyjny przyznał procesowi użytkownika, a drugi jego rozmiar.

# Prosta ochrona pamięci

Każdy adres wygenerowany przez proces użytkownika sprawdzany jest zgodnie ze schematem umieszczonym na następnej planszy. Jeśli test nie powiedzie się uruchamiana jest pułapka i sterowanie wraca do systemu operacyjnego. Rozkazy modyfikujące rejestr bazowy i graniczny muszą być uprzywilejowane.

# Prosta ochrona pamięci



# Czasomierz

Opisane wcześniej mechanizmy nie chronią systemu przed sytuacją, w której proces użytkownika przejmuje procesor i nigdy go nie oddaje (np.: na skutek wystąpienia niekończącej się pętli). Taką ochronę może zagwarantować sprzętowy licznik czasu. Jest on najczęściej implementowany jako licznik zliczający w dół, którego stan zmienia się co takt zegara. Po wyzerowaniu licznika sterownie wraca do systemu operacyjnego. Opisany czasomierz (ang. *timer*) może służyć jako układ typu *watchdog*, który np.: w systemie czasu rzeczywistego pozwalałby sprawdzić, czy zadanie wykonało się w przeznaczonym dla niego czasie. W systemach z podziałem czasu ten układ wyznacza moment, w którym system operacyjny powinien przełączyć procesor na inne zadanie użytkownika. Może on również służyć do (niezbyt dokładnego) wyznaczania pory dnia. Inicjacji czasomierza dokonuje system operacyjny, przed oddaniem sterowania procesowi użytkownika. Rozkaz inicjacji jest rozkazem uprzywilejowanym.

# Wywołania systemowe

Skoro tylko system operacyjny może wykonywać operacje I/O i szereg innych ważnych czynności, to w jaki sposób proces użytkownika może się komunikować ze światem zewnętrznym lub podejmować inne działania konieczne do jego realizacji? Otóż współczesne komputery udostępniają procesom użytkownika rozkaz[2], który umożliwia zażądanie od systemu operacyjnego wykonania wymaganych przez nie operacji. Ten rozkaz jest nazywany przerwaniem programowym, ponieważ powoduje przełączenie procesora w tryb monitora, oraz wykonanie odpowiedniej procedury obsługi przerwania. Ta procedura realizuje zamówioną przez proces użytkownika czynność. Nazywana jest ona *wywołaniem systemowym (ang. system call)* lub *funkcją systemową*. Proces użytkownika może przekazać jej pewne argumenty, ale zanim zostaną one użyte, muszą być przez procedurę dokładnie sprawdzone. Podsumowując — przerwania programowe umożliwiają procesom użytkownika korzystnie z usług systemu operacyjnego.

[2] W procesorach Intel jest to rozkaz INT.

# Systemy wieloprocesorowe

Systemy wieloprocesorowe (równoległe) mają wiele zalet. Do największych z nich należy oczywiście wydajność. Należy jednak pamiętać, że system złożony z N procesorów nie wykonuje zadań N-razy szybciej niż system jednoprocesorowy. Przyspieszenie może być proporcjonalne do tej wartości, ale nie równe. Inną zaletą jest niezawodność — jeśli system składa się z takich samych procesorów, to w razie awarii jednego z nich, jego obowiązki przejmują pozostałe. Jeśli procesory są różne, takie rozwiązanie nie jest możliwe. Popularnym rozwiązaniem jest symetryczne wieloprzetwarzanie (ang. *SMP*). Polega ono na tym, że w systemie umieszczonych jest kilka jednakowych procesorów, a każdy z nich wykonuje własną kopię oprogramowania systemowego i przydzielone mu procesy użytkowników. Przed systemem operacyjnym stoi zadanie koordynacji pracy tych procesorów. We wczesnych systemach komputerowych popularne było wieloprzetwarzanie asymetryczne. W takim schemacie jeden, uprzywilejowany procesor zajmował się przetwarzaniem danych, pozostałe operacjami I/O. W chwili obecnej każdy komputer jest wyposażony w kontrolery, które zapewniają taki rodzaj wieloprzetwarzania, ale to określenie już nie funkcjonuje.

# Architektury komputerów osobistych

Pierwsze mikroprocesory, takie jak Intel 8086, czy Motorola MC68000 pozbawione były środków sprzętowych wymaganych do zapewnienia choćby minimalnej ochrony systemowi operacyjnemu i procesom użytkownika. Rzutowało to oczywiście na budowę i sposób pracy pierwszych systemów operacyjnych przeznaczonych na te komputery (MS-DOS, MacOS). Z czasem pojawiły się droższe od komputerów klasy IBM PC stacje robocze (SUN, Apollo, Silicon Graphics), których procesory zapewniały odpowiednie mechanizmy ochrony. Te mechanizmy zaczęły się pojawiać w tańszych rozwiązaniach, co umożliwiło pisanie bardziej bezpiecznych i stabilnych systemów operacyjnych lub przenoszenie na komputery osobiste systemów znanych z „dużych" systemów komputerowych. Przykładem rodziny procesorów, która przeszła ewolucję od urządzeń w ogóle pozbawionych środków ochrony do urządzeń z elastycznym systemem ochrony jest rodzina x86. Obecnie procesory te potrafią pracować w czterech trybach nazywanych pierścieniami (ang. *ring*), przy czym najbardziej uprzywilejowany jest pierścień zerowy. Ten mechanizm wzorowany jest na systemie Multics.

# Pytania

?

# Koniec

Dziękuję Państwu za uwagę.