Obsługa We/Wy

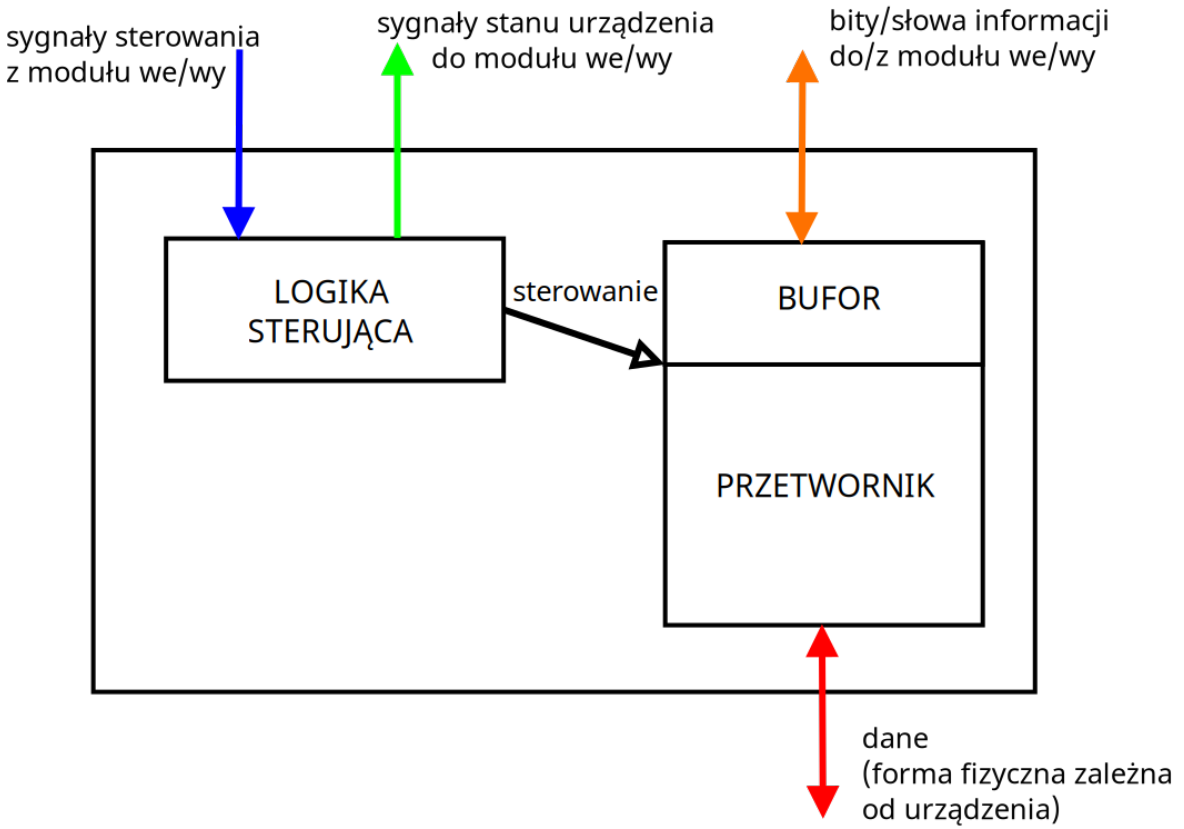
Przypomnijmy...

- Komponenty komputera (Neuman)
  - procesor, pamięć, układy we-wy
- Magistrala systemowa
  - adresy, dane, sterowanie
- Różne szybkości
- Różne formaty danych

Urządzenia peryferyjne

- Do odczytu przez człowieka
- Do odczytu przez sprzęt
- Do komunikacji
- Są szybsze/wolniejsze od procesora
- Przesyłają pojedyncze dane/bloki danych

Urządzenie zewnętrzne



Moduł we/wy – funkcje

- Sterowanie i synchronizacja
- Komunikacja z procesorem i urządzeniem
- Buforowanie danych
  - wyrównywanie szybkości CPU-urządzenie
- Wykrywanie (i obsługa) błędów
  - błędy mechaniczne
  - błędy logiczne (transmisji, parzystości...)

Sterowanie/synchronizacja

- Wiele urządzeń
- Losowy schemat dostępu do we/wy
- Jedna magistrala

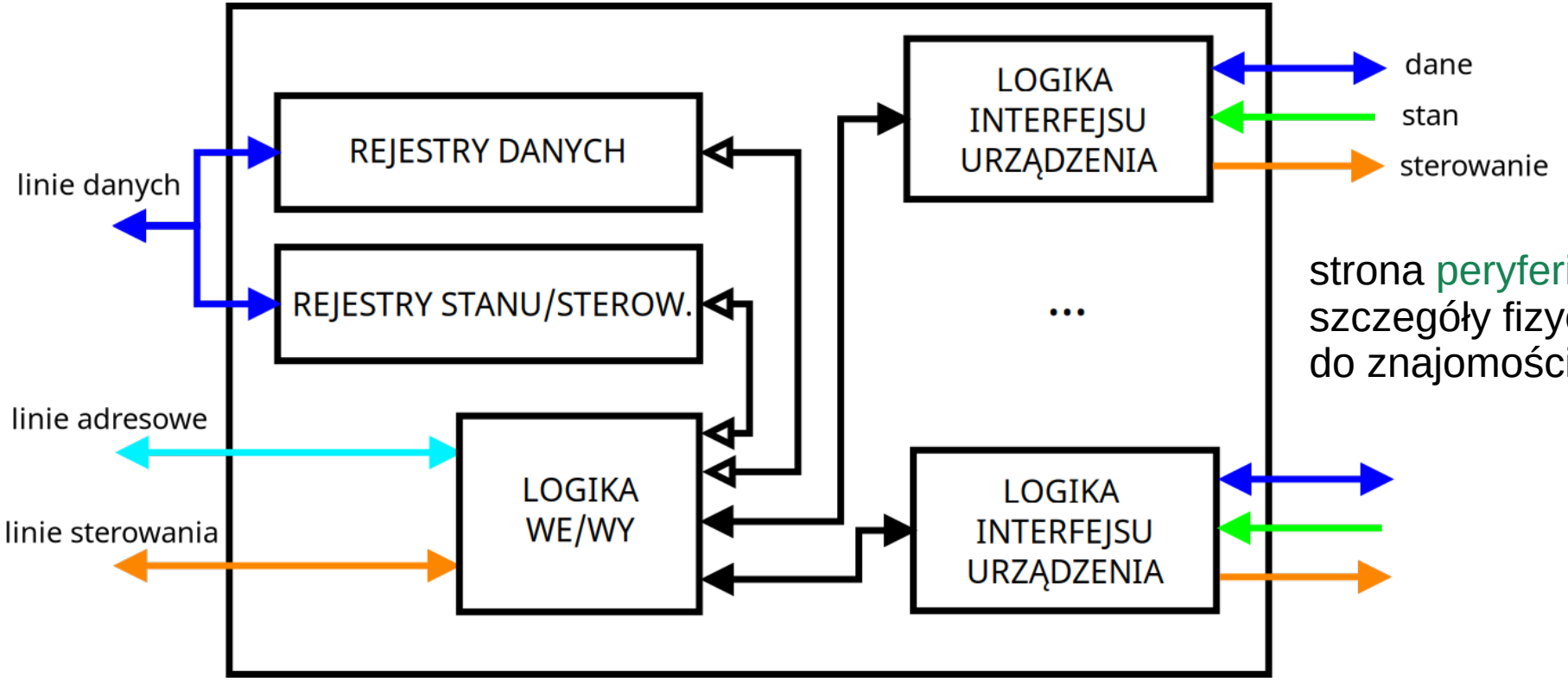
1. CPU: sprawdza stan U
2. MIO: U jest dostępne
3. CPU: żądanie danych (rozkaz do MIO)
4. MIO: pobiera dane z U
5. MIO: oddaje dane do CPU

Komunikacja z CPU

- Dekodowanie poleceń
  - czytaj sektor, czyść ekran, flush() ...
- Przesyłanie danych
- Przesyłanie stanu
  - obecność, gotowość, zakończenie operacji, błąd...
- Rozpoznawanie adresów
  - adresy kontrolowanych peryferiów (i swoje)

Moduł we/wy

strona procesora
jednolity interfejs



strona peryferii
szczegóły fizyczne
do znajomości modułu

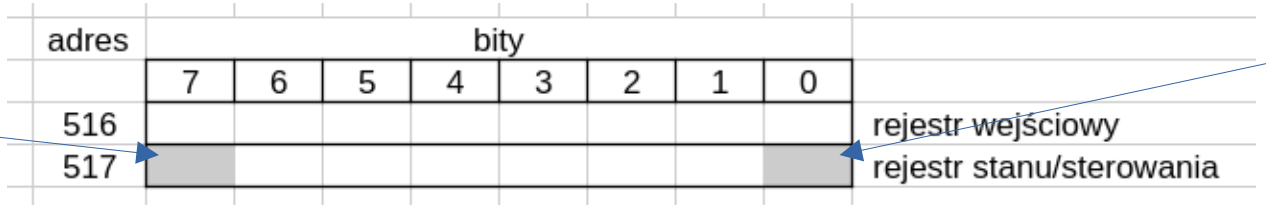
Programowane we/wy

- Procesor *rozkazuje* modułowi we/wy
  - sterowanie, testowanie, odczyt, zapis
- Procesor utrzymuje cały czas kontrolę nad operacją we/wy (inicjuje, czeka, kończy)
  - MIO realizuje operację i ustawia rejestry statusu
  - CPU zagląda do rejestrów i reaguje odpowiednio

Rozkazy we/wy

- Bliska jednoznaczność instrukcji programu z rozkazem dla MIO
- Peryferia mają adres, podawany na szynę
  - we/wy odwzorowane w pamięci (memory-mapped IO)
    - cała przestrzeń adr. dotyczy pamięci i peryferiów
  - we/wy odizolowane
    - adres peryferiów wskazywany osobną linią

1 – gotowy
0 – zajęty



1 – start
0 – idle

ADRES	ROZKAZ	ARGUMENT	OPIS
200	LOAD AC	1	ładuj „1” do akumulatora
	STORE AC	517	zacznij odczyt klawiatury
202	LOAD AC	517	czytaj bajt stanu
	BRANCH if Sign == 0	202	czekaj na „1”
	LOAD AC	516	odczytaj kod klawisza

ADRES	ROZKAZ	ARGUMENT	OPIS
200	LOAD IO	5	rozpocznij odczyt klawiatury (urz. #5)
	TEST IO	5	sprawdź status operacji czytania (urz. #5)
202	BRANCH NRDY	201	czekaj na gotowość
	IN	5	odczytaj kod klawisza

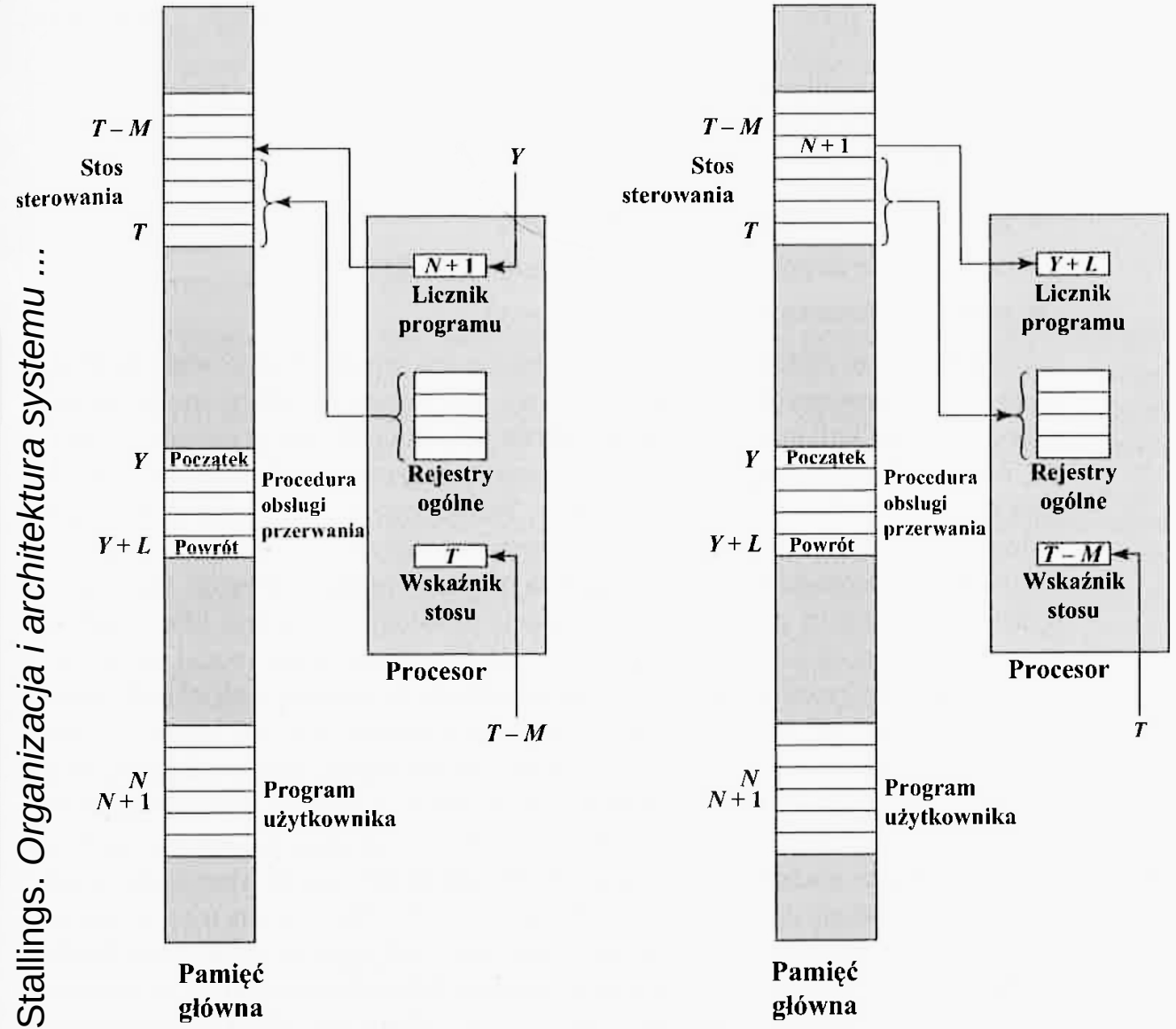
We/wy z przerwaniami

- Procesor inicjuje operację
  - … i robi coś innego
  - na końcu cyklu rozkazu sprawdza przerwanie
- MIO realizuje operację i sygnalizuje koniec
- Procesor żąda danych
  - dane trafiają na magistralę
  - MIO gotowy do dalszych operacji

Obsługa przerwania

1. Urządzenie wysyła żądanie przerwania (IRQ)
2. CPU kończy rozkaz, sprawdza przerwania
3. Stwierdza przerwanie, potwierdza
4. Zrzucenie stanu (stan procesora – PSW, PC) na stos
5. Załadowanie do PC adresu pierwszego rozkazu obsługi przerwania
6. Zapisanie zawartości wszystkich rejestrów na stosie
7. Obsługa przerwania, realizacja we/wy itp.
8. Odtworzenie przerwanego programu ze stosu
9. Odtworzenie PSW, PC przerwanego programu

Obsługa przerwania działania na stosie



Stallings. *Organizacja i architektura systemu ...*

Wiele przerwań

PRIORYTETY?

- Wiele linii przerwań
- Odpytywanie programowe
  - ogólna obsługa przerwania odpytuje wszystkie MIO
- Odpytywanie sprzętowe
  - wspólna linia **IRQ**, urządzenia łańcuchowo na linii **IACK**
  - urządzenie odbierające **IACK** wystawia swój wektor na magistrali danych
- Arbitraż magistrali
  - przed zgłoszeniem żądania urządzenie zajmuje magistralę
  - po odebraniu **IACK** od CPU wystawia swój wektor na magistralę

Bezpośredni dostęp do pamięci (DMA)

- Dotychczas procesor musiał aktywnie działać podczas przesyłu
- Przeładowanie kontekstu do obsługi żądania to czas
- Testowanie i obsługa urządzenia jest długa

- Programowe we/wy: dobre dla bloków danych
- we/wy z przerwaniami: relatywnie szybkie

Moduł DMA

- Przejmuje sterowanie magistralą
- Przesyła dane na magistrali, gdy CPU „liczy”
  - lub „zawiesza” CPU przy operacjach na magistrali („*cycle stealing*”): ARG/DATA FETCH, STORE
- Zgłasza koniec operacji (przerwanie)
- Np. Intel 8237A

DMA a pamięć cache – przykład sieciowy

1. **Nadchodzi** pakiet (NIC zdejmuje „ethernet” zostaje TCP/IP) – oddane do DMA
2. DMA przesyła dane do pam. gł. → unieważnia cache
3. NIC wywołuje przerwanie (pakiet gotowy)
4. Rdzeń pobiera TCP/IP → chybienie w cache
5. Przetwarzanie nagłówków (zajrzenie do TCB → możliwe chybienie)
6. Odebranie zawartości, przesłanie do bufora aplikacji (dane są już w cache)

DMA a pamięć cache – przykład sieciowy

1. Pakiet gotowy **do wysłania** (wyw. systemowe)
2. OS uruchamia proces stosu TCP/IP (TCB → możliwe chybienie), kopiuje dane z pam. procesu do pam. syst. (teraz dane są w cache!)
3. Wywołanie sterownika, który uruchamia DMA
4. Transfer DMA z cache do NIC (+unieważnienie linii)
5. NIC potwierdza wysłanie
6. Sterownik karty czyści bufor, gotowy na następne dane