SYSTEMY WBUDOWANE 8^{00} - 10^{55}

prof. KRZYSZTOF SACHA pok 42 14^{25} - 15

Sala 210

Wykład 1 2000/02/22

System czasu rzeczywistego (real time system) jest to system komputerowy w którym obliczenia komputerowe prowadzone równolegle z przebiegiem zewnętrznego procesu mają na celu monitorowanie, sterowanie lub terminowe reagowanie na zdarzenia w tym procesie zdarzenia

System wbudowany (embedded system) jest to system komputerowy będący częścią większego systemu i wykonujący istotną część jego funkcji, przykładem może być komputer pokładowy samolotu lub system sterujący szybkiej kolei miejskiej

Charakterystyka systemów

- połączenie z procesem zewnętrznym
- ograniczenie czasowe np. czas reakcji systemu
- sterowanie zdarzeniami z przeliczeniem priorytetów

Sterowanie przejazdem kolejowym

zdarzenie

a) pociąg się zbliża do przejazdu

Reakcje

- zamknij rogatkę, wyświetli się zielone światło

b) pociąg odjeżdża

- wyświetl czerwone światło, otwórz rogatkę

Ograniczenie czasowe:

zamknięcie w czasie $< \Delta t_1$ (bo katastrofa)
wyświetlenie w czasie $< \Delta t_2$ (bo hamowanie)

czas

$$t = \frac{s}{v}$$

s → droga, v → prędkość

$$t = \frac{s}{v} = \frac{4km}{200\,km/g} = \frac{1}{50}\,godz \approx 1\,min$$

ŻYROSKOP

1) sensory przyspieszeń kątowych (czujniki) (3D)
2) sensory przyspieszeń liniowych (3D)

} odbiór sensorów musi być co 10 ms

S - droga
V - prędkość
a - przyspieszenie

$$V = \frac{ds}{dt}$$

ds → pochodna drogi, dt → pochodna czasu

$$a = \frac{dv}{dt}$$

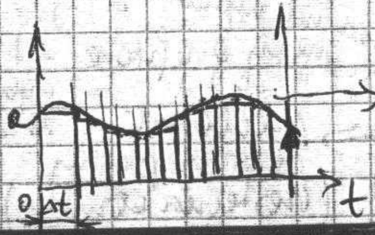
dv → pochodna prędkości, dt → pochodna czasu

$$V = \int_{t=0}^{t} a\,dt$$

$\int$ - całka

$$S = \int_0^t v\,dt \approx a_0 \Delta t + a_1 \Delta t + \dots a_n \Delta t = \sum_{i=0}^{n} a_i \Delta t$$

bierzemy działki czasowe



→ jest to pole pod krzywą

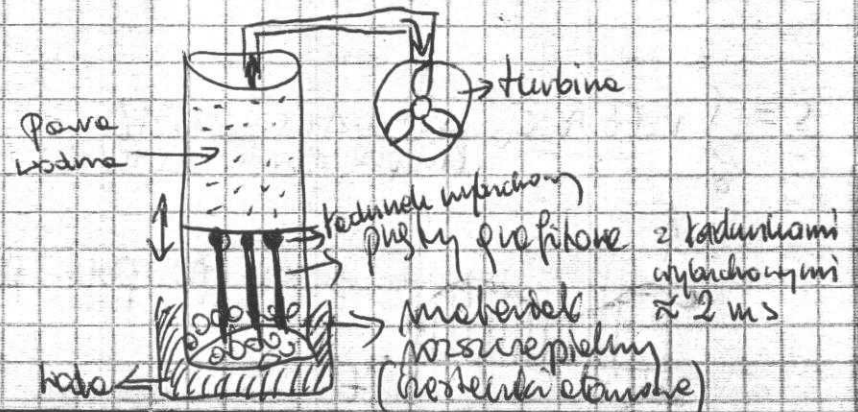
Całkowanie jest sygnałem zewnętrznym (przerywane)

wyświetlanie co 40ms
3) całkowanie przyspieszeń żeby uzyskać prędkość
4) całkowanie prędkości żeby uzyskać drogę

co 1s 5) Kompensacja uchybów (zróżnicowanie błędów)

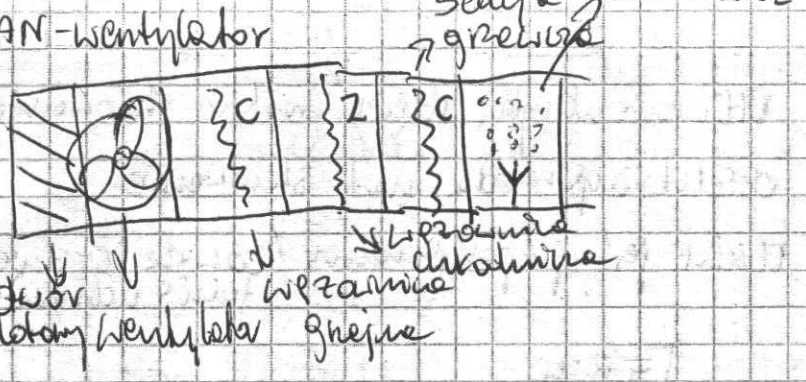
co 40ms 6) Wyświetlanie wyników na lokalizacji

Nadzór sikomm jednawej
- Alarmy kontroli dostępu - strefy bezpieczeństwa odpowiedź na sygnał naruszenie kontroli nie większym niż ≤ 1 sek
- Alarm przepięcia rezultat - odpowiedź w czasie mniejszym niż ≤ 1 ms
- Wyświetlenie na pulpicie w czasie co 33ms

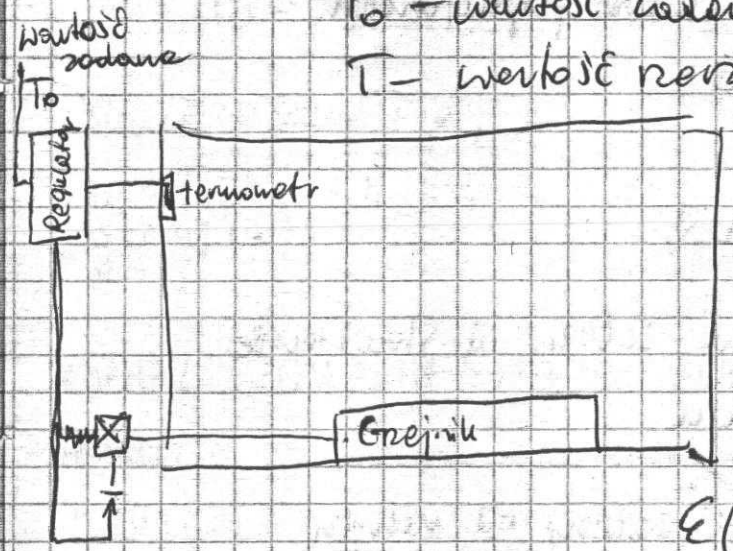


ATRIUM TOWER - SACHA ZROBIŁ PROJEKT KLIMATYZACJI

KLIMATYZACJA



T_0 - wartość zadana
$\bar{T}$ - wartość rzeczywista



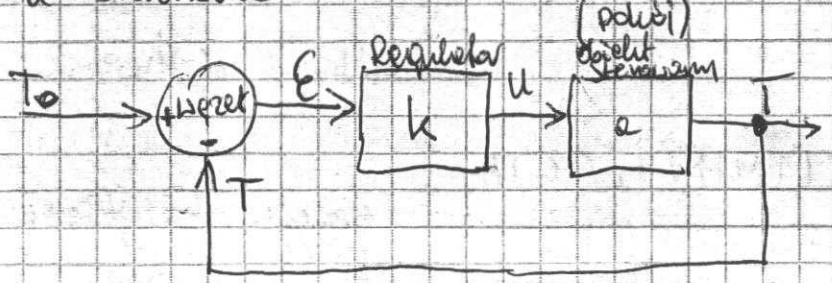
Uchyło (błąd) $\varepsilon = T_0 - \bar{T}$

wymagane: $\varepsilon \leq 1°C$

(gdzie to jest nas?)

REGULATOR PROPORCJONALNY typu P

u - sterowanie



Obliczenie uchybu, obliczenie wartości sterowania obiekt otrzymuje nowe sterowanie

- obiekt jest proporcjonalny (rośnie będzie jeśli uchyb)

$$T = a \cdot u$$

- Regulator działa proporcjonalnie

$$u = k \cdot \varepsilon$$

- Uchyb

$$\varepsilon = T_o - T$$

- temperatura zależy od sterowania

$$\varepsilon = T_o - a \cdot u$$

- Sterowanie zależy od uchybu

$$\varepsilon = T_o - a \cdot k \cdot e$$

$$\varepsilon \cdot (1 + a \cdot k) = T_o$$

$$\varepsilon = \frac{T_o}{1 + a \cdot k}$$

REGULATOR PROPORCJONALNO-CAŁKUJĄCY typu PI

$$u = k_1 \cdot \varepsilon + k_2 \cdot \int_0^t \varepsilon \, dt$$

$$\approx k_1 \cdot \varepsilon + k_2 \cdot \underbrace{\sum_{i=0}^{n} \varepsilon_i \Delta t}$$

$$\varepsilon_0 + \varepsilon_1 + \varepsilon_2 + \ldots + \varepsilon_n$$

Powtarzaj co czas Δt {

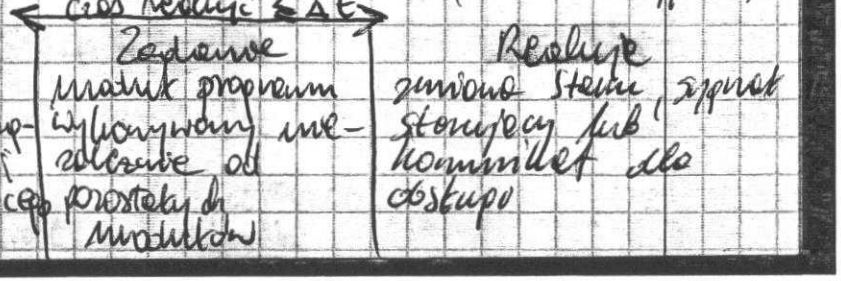
```
T = pomiar Temp ( );   // pomiar temperatury
e = To - T;            // obliczenie uchybu

Suma = suma + e;

u = k1 * e + k2 * suma * dt;
Steruj (u);            // sterowanie grzejnikiem
end;
```

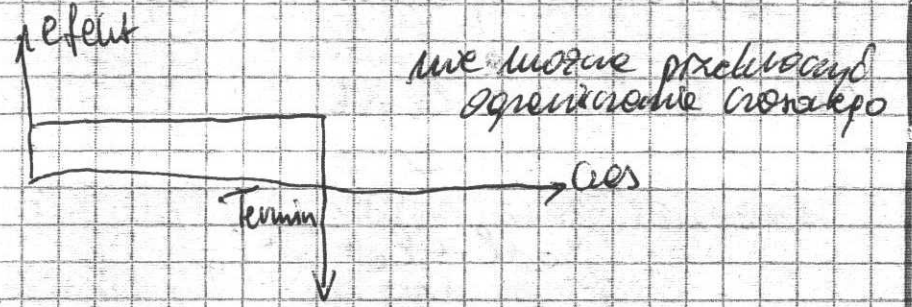
MODEL DZIAŁANIA SYSTEMU WBUDOWANEGO

- działanie reaktywne zdarzeniami (event triggered)
- działanie reaktywne czasem (time triggered)

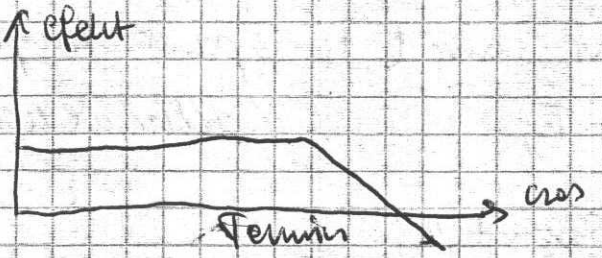


KLASYFIKACJA

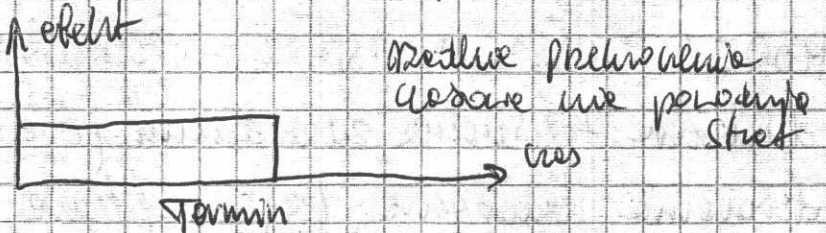
- Ostre ograniczenie czasowe (hard real time)



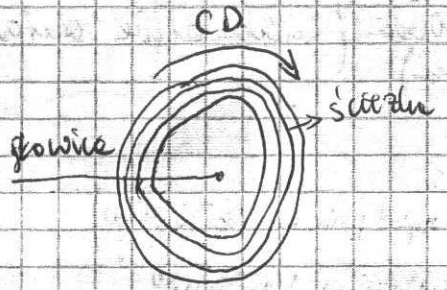
- Łagodne ograniczenie czasowe (soft real-time)

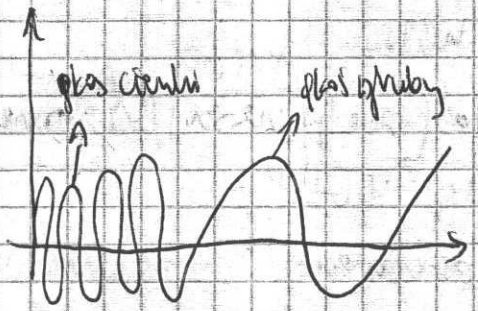


- Sztywne ograniczenie czasowe (firm real-time)



Zapisanie dźwięku na CD





WYKŁAD 2 2009/03/08

OPRACOWANIE SYSTEMU

- specyfikacja wymagań
- określenie zadań
- projekt sprzętu
- określenie modelowania wymagania (określenie zadań)
- implementacja
  C, assembler
- weryfikacja projektu
  poprawność funkcjonalna
  terminowość wykonania

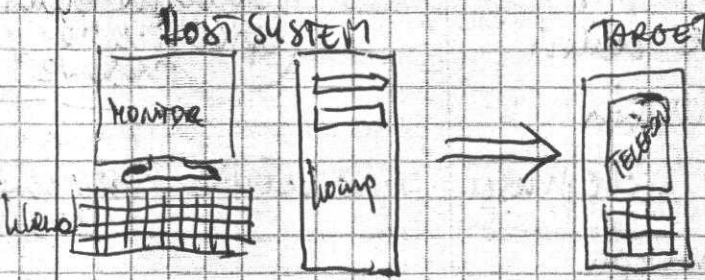
zadanie mają ograniczenia czasowe

pre-run scheduling (wykonanie ręczne)
run-time scheduling (wykonanie automatyczne)
(wykonanie przed czasem)

C++ (klasy, obiekty, współpraca obiektów)
C - zawiera 75% modułów w systemie
assembler - zawiera 40%
C++ - zawiera 40% (nie w części RT, raczej w GUI)
JAVA - 5% (eksperyment)

ASSEMBLER (język maszynowy)

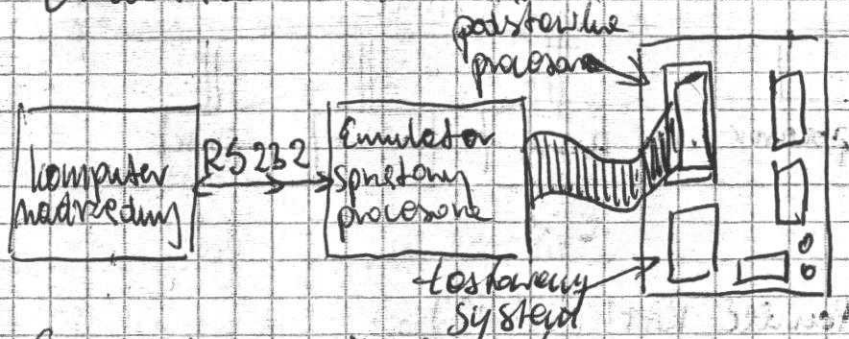
ŚRODOWISKO SKROŚNE



- kompilator skrośny (działa na jednym systemie a tłumaczy na inny język)
- Symulator - specjalny program
- Emulator układowy

EPROM - ELECTRICALY PROGRAMMABLE READ ONLY MEMORY

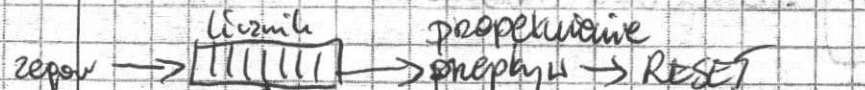
Symulator
EMULATOR UKŁADOWY

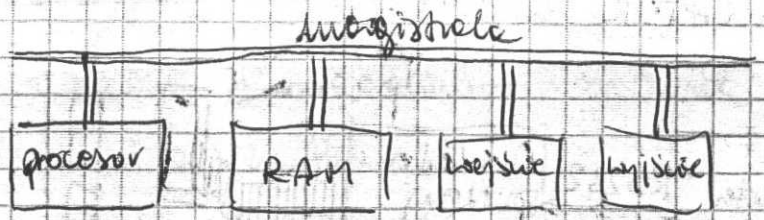


- Emulator - mikrokontroler z programem monitor (rejonuje część zasobów: pamięć, port szeregowy)
- komputer nadrzędny - program uruchomieniowy (pułapki, proces śledzony, modyfikowanie stanu)

STEROWNIK WBUDOWANY (sprzęt procesowy interfejs)
wymagania
- układy wejść - wyjść procesowych
- odporność na trudne warunki pracy
- możliwość pracy ciągłej
- niezawodność działania - oprogramowany pobór mocy

WATCHDOG - BUDZIK





Pamięć RAM (1024 kB)

$2^{32} = 2^{30} \cdot 2^2 = 4 \cdot 2^{30} = 4(2^{10})^3 = 4(1024)^3 =$
$= 4(10^3)^3 = 4 \cdot 10^9$

0. Jeśli przerwanie to → przejdź do innego procesu
1. Odczytaj instrukcję z komórki pamięci wskazanej przez licznik rozkazów
2. Zwiększ zawartość licznika rozkazów
3. Wykonaj instrukcję → ostrzeż sprzętowy

---

a = b + c

x = a ↓ 3

if (x > 0) a = 5

Można zmienić adres pamięci

a	2
b	6
c	10
x	14

1. kopiuj z adresu 6 do R1
2. — " — " — 10 do R2
3. dodaj R1, R2, R3
4. Pamiętaj R3 pod adresem 2
   Porównaj 14 z wartością ≠ 0
   Skocz jeśli mniejsze M
   pamiętaj 2, ≠ 0

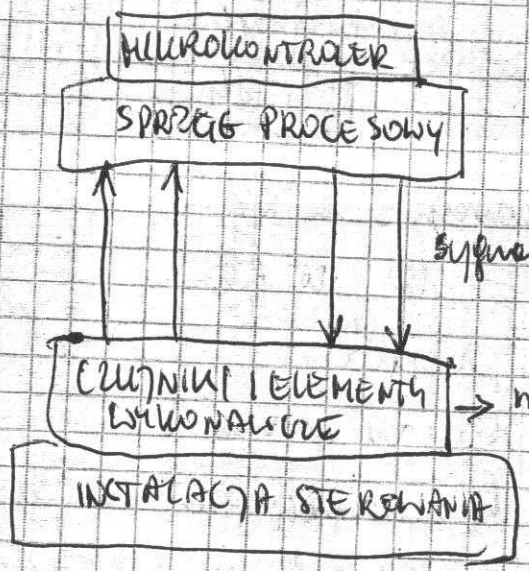
KOMPUTER KASETOWY (np. VME)

- Układy wejść wyjść procesorych
- odpowiedzialne za tworzenie warunków pracy.

1. Sczytaj licznik rozkazów
2. Idź do licznika rozkazów adres programu obsługi przerwania do programu

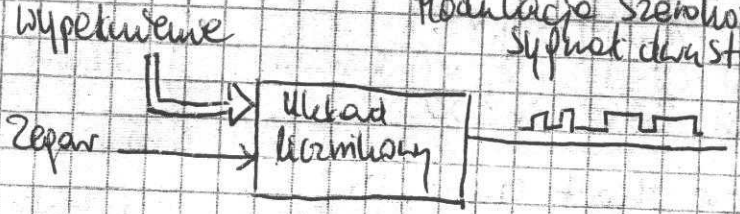
zawartość licznika rozkazów

SPRZĘG PROCESOWY (process interface)



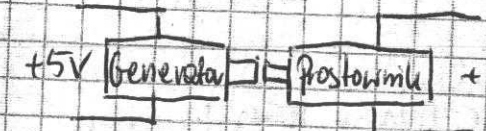
Sygnały analogowe	Sygnały dwustanowe
4-20 mA (0-20 mA)	0-24V / 0,5A
0-10V	styk
specjalne	0-5V (pomocnicze)

STEROWANIE PWM (Pulse Width Modulation – Modulacja Szerokości Impulsu)

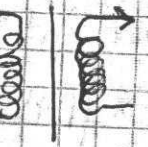


IZOLACJA GALWANICZNA

Niezależność zasilania



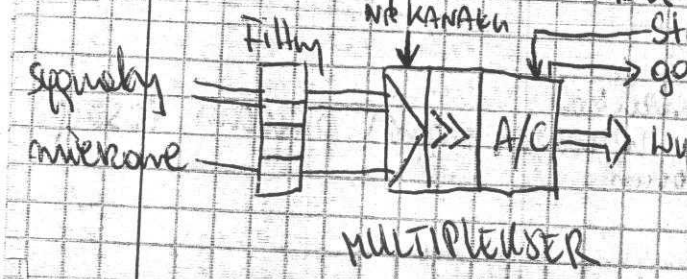
Transformator – są to dwie cewki (przetwornice)



WYKŁAD 3

2008/03/22

WIELOKANAŁOWY PRZETWORNIK A/C (analogowo-cyfrowy)



Karta interfejsowa PCL 718

DO – wyjście cyfrowe (digital output)

DMA – DIRECT MEMORY ACCESS

BEZPOŚREDNI DOSTĘP DO PAMIĘCI

Odczyt karty PCL 718

```
#include "pcl718.h"
#define BASE 0x300
unsigned short czytajDi()
{
  short int dl, dh;
  dl = in8(BASE+3);
  dh = in8(BASE+M);
   return (dh<<8)+dl;
}
```

```
#include „pcl718.h"
#define BASE 0x300
unsigned short czytajAC(char kan)
{
  short int stan, al, ah, i;
  out8(BASE+2, (kan<<4)|kan);
  out8(BASE, 0);
  do {
  stan = in8(BASE+8);
    i++;
  if (i>10000) return -1;
}
  while ((stand & 0x80) != 0);
  ah = in8(BASE+1);
  al = in8(BASE+0);
  return (ah<<4)+(al>>4);
}
```

UZYCIE PRZERWAN

```
#define BASE 0x300
#define INTR 3
struct sigevent event
void intAC() {
  out8(BASE+9, 0x80)|(INTR<<); // ustaw przerwanie
  InterruptAttach(INTR, &handler); // instaluj handler
}
unsigned short odczytAc(int kan) {
  char al, ah;
out8(BASE+2, kan<<4|kan); // zalacz multiplekserc
out8(BASE+0, 0);          // start pomiaru
InterruptWait();
ah = in8(BASE+1);
```

```
    al = in8 (BASE+0);
    return (ah<<4) + (al>>4);
}
struct sigevent *handler() {
    out8(BASE+8, 0);        //skasuj przerwanie
    return &event;
}
```

INTERRUPT - PRZERWANIE

Komputer jednopłytkowy lub piggyback (np. PC/104)

Komputer kasetowy (np. VME)

Komputer panelowy (Panel PC)

- procesor bezwentylatorowy
- RAM
- Ekran dotykowy (15", 17")
- Ethernet (10M, 100M, 1G)
- 2 porty USB
- 2 porty PS/2
- 4 porty szeregowe
- port równoległy

WYKŁAD 4 2009/04/05

PROCESORY WBUDOWANE

- 80C51 (INTEL) - 8 bitowy CISC
- x86 (INTEL) - 16 i 32 bitowe CISC
- 68K (Motorola) - 32 bitowe CISC
  - PowerPC
  - MPC8xx, MPC5xx
- ACORN COMPUTERS (ARM) - 32 bitowe RISC
  - Strong ARM (DEC)
  - xSCALE (Intel)
- MIPS (MIPS TECHNOLOGIES) - 32 i 64 bitowe RISC
- SH (HITACHI) - 32 i 64 bitowe RISC

8048 - komputer jednoukładowy

8051 -

RISC - Reduce

CISC - Complex

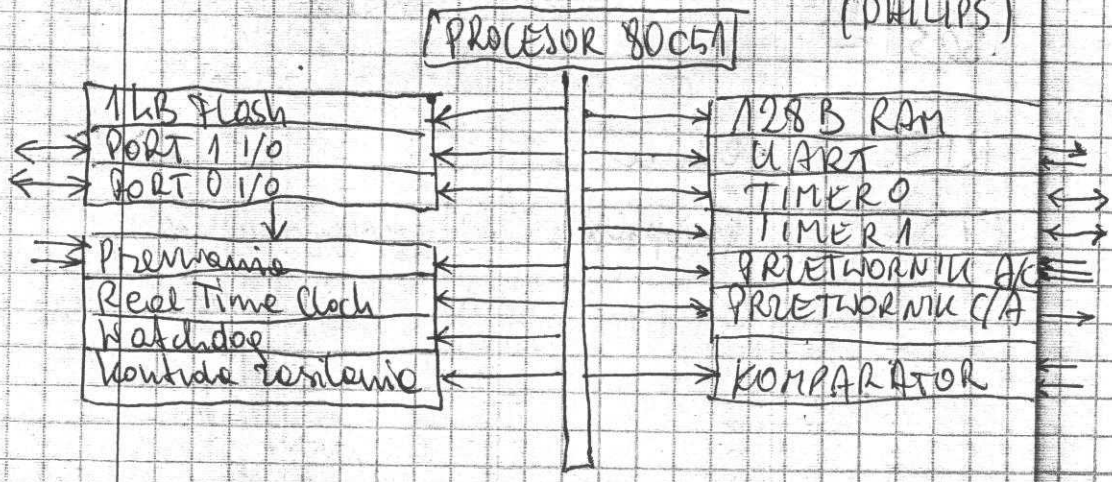
MIKROKONTROLER 80C51

- 8 bitowy procesor CISC
- dwuporziomowy układ przerwań
- procesor binarny
- tryb oszczędzania energii
- 4 kB pamięci ROM, 128 B pamięci RAM
- 4 dwukierunkowe rejestry wej/wyj
- 1 port szeregowy (UART)
- 2 programowane liczniki czasu (timer)

ROM - dla pamięci programu (4kB)

RAM - pamięć danych (128 Bajtów)

PRZYKŁAD MIKROKONTROLERA P89LPC9107 (PHILIPS)

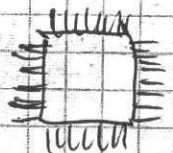


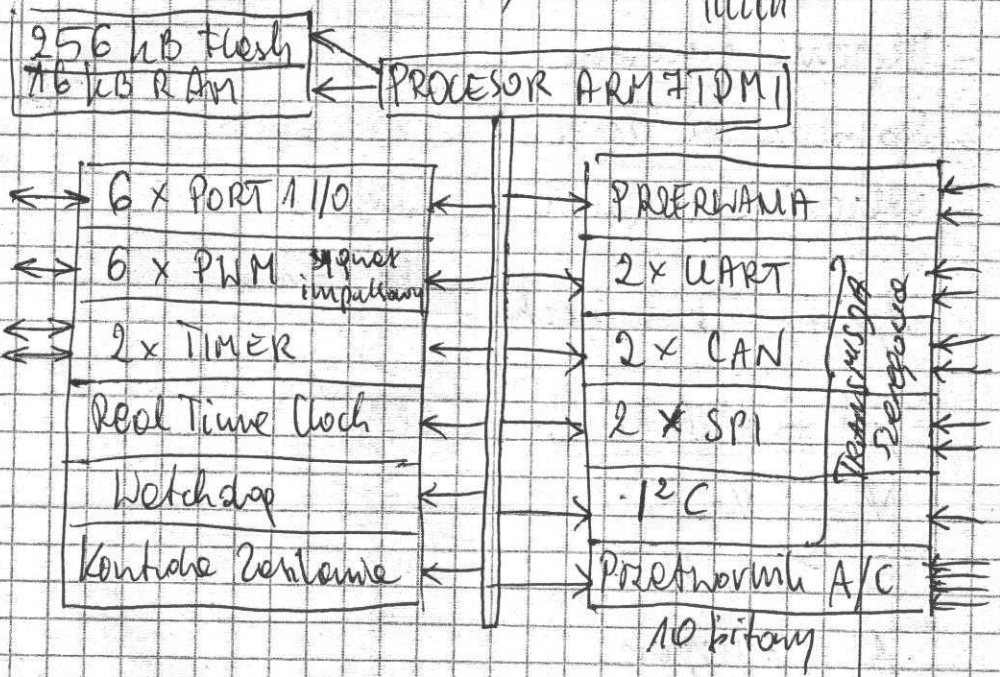
UART (Universal Asynchronous Receiver Transmiter) port szeregowy

KOMPARATOR (COMPARE) - porównywać

A/C - analogowo - cyfrowe

MIKROKONTROLER LPC2128 (PHILIPS)





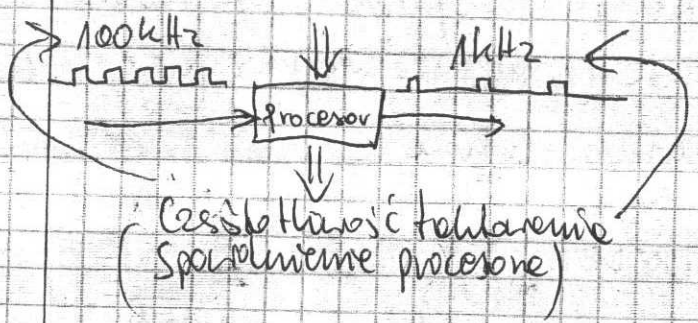
CAN - norma sieci przemysłowej

wprowadzona w branży samochodowej

(Mercedes) ciężkiej, jest w 4,5%

SPI - przyrządy pomiarowe, laboratorium

I²C – jest port USB



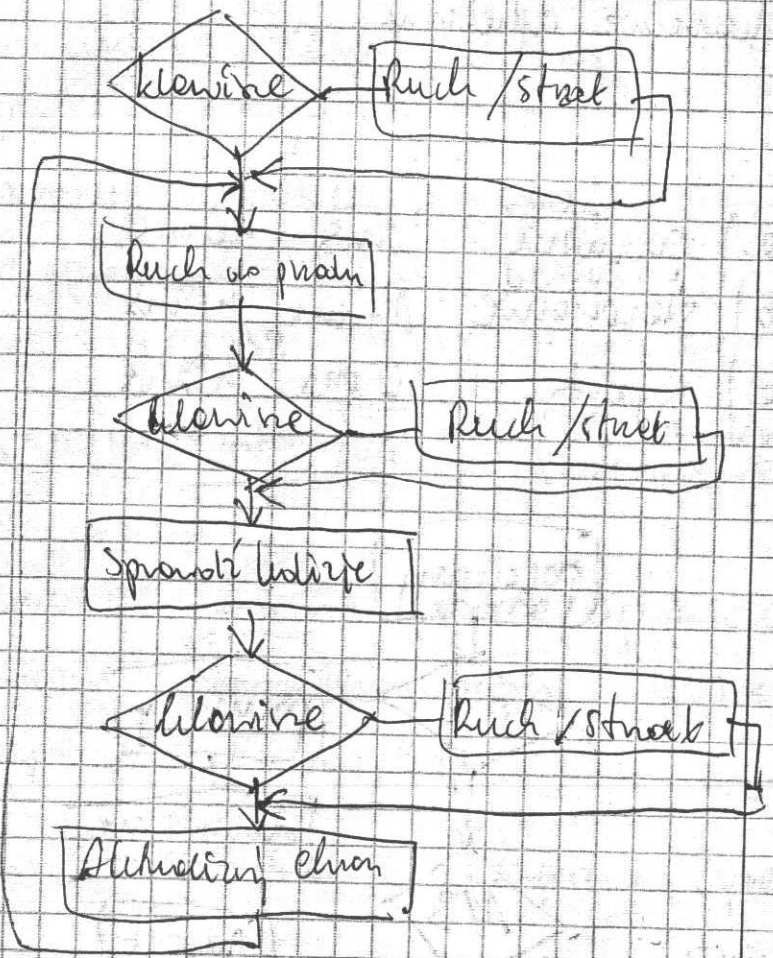
- Uśpienie procesora
- Spowolnienie zegara
- Odłączenie układów zewnętrznych

1μA × 5V = 5μW

μ – mikro
V – Volt
W – Watt
A – Amper

Transmisja szeregowa (komunikacja)

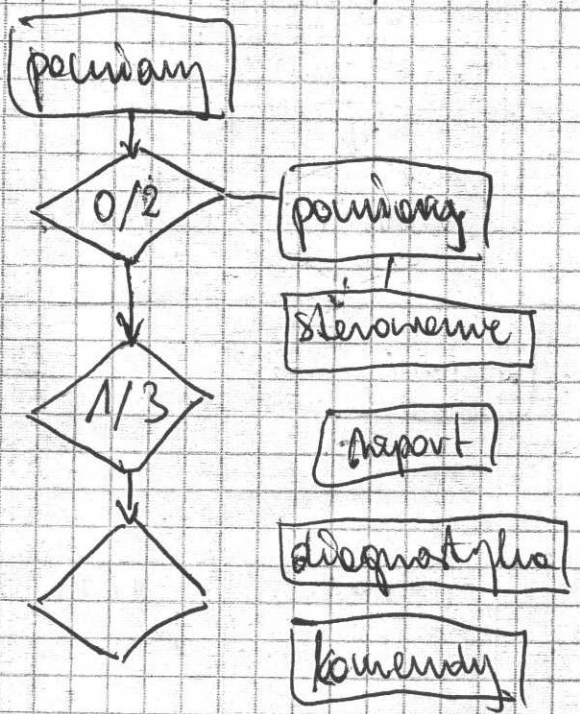
ODBLOKOWANIE DRZWI



WYKŁAD 2021/04/26

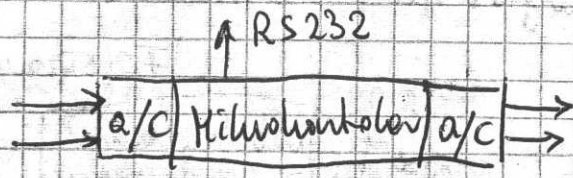
Wywłaszczanie aplikacyjne

(a)	pomiary	2 ms	40 ms
(b)	sterowanie	14 ms	40 ms
(c)	raport	10 ms	40 ms



System sterowania (foreground / background)

		(FPLAN)	(TKO)
(a)	pomiary	2 ms	40 ms
(b)	sterowanie	14 ms	40 ms
(c)	raport	50 ms	500 ms
(d)	diagnostyka	30 ms	500 ms



Zadanie krytyczne (ostre ograniczenie)
- związane ze sterowaniem - czasowe
- bezpieczeństwo instalacji

Zadanie pomocnicze (łagodne ograniczenie - czasowe)

Okres 500 ms w tym: 500/40 ≈ 13

≈ 13 przerwań (40 ms) →

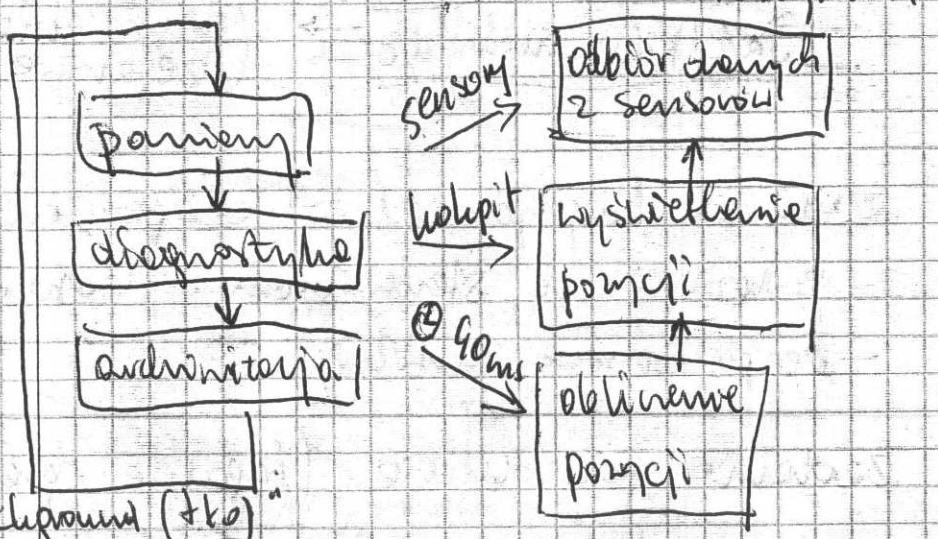
1) 13 × 16 ms = 208 ms

2) 500 − 208 = 292 ms

Wykluczanie przerwań

Bezkolejnościowy system wieloprocesowy

(a) odbiór wskazań sensorów co 10ms (tak/awaria)
(b) obliczenie prędkości pozycji co 40ms
(c) wyświetlenie pozycji co 40ms (tak/awaria)
(d) pauzy, diagnostyka, archiwizacja



Wielozadaniowy system operacyjny

- Dużo zadań bez regularnego schematu wykonania
- Zadania z dużymi wymaganiami

- komunikacja w sieci
- drivery urządzeń
- GUI
- archiwizacji i dokumentacja
- Szeregowanie zadań (run-time scheduling)
- Synchronizacja i komunikacja zadań

Szeregowanie zadań

Sformułowanie problemu

Dany zbiór zadań $\{ z_i : z_i = (t_i, c_i) \}$ (zadanie, czas wykonania, okres wykonywania, termin zakończenia)

- Znaleźć taką kolejność wykonania zadań, przy której każde zadanie wykona się w ramach wymaganego okresu
- Od czego zależy możliwość wykonania zadań w terminie 1) od obciążenia

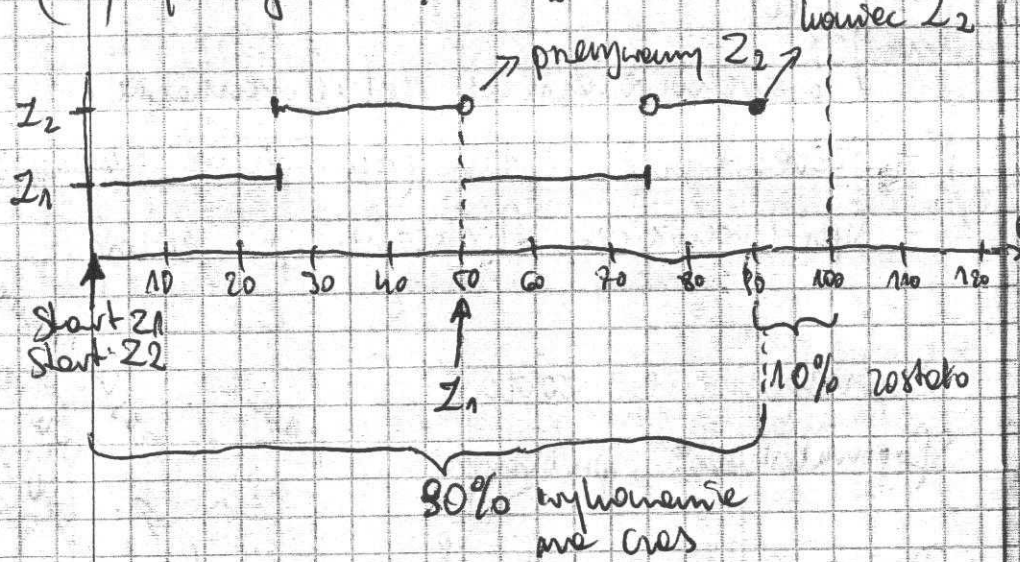
przykład: $\sum_i \frac{t_i}{c_i}$ ← obciążenie procesora przez zadanie z_i

$z_1 = (25ms, 50ms)$
$z_2 = (40ms, 100ms)$

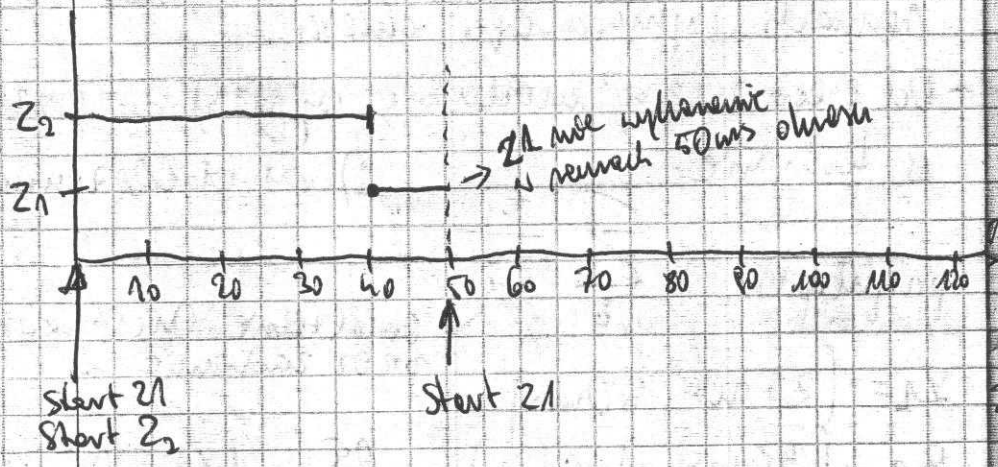
$\frac{25}{50} + \frac{40}{100} = 0,5 + 0,4 = 0,8 = 80\%$

2) od kolejności? odp: tak zależy

(a) priorytet $Z_1 > Z_2$



(b) priorytet $Z_2 > Z_1$



3) czy zależy od konkretnego układu wartości t_i, c_i?

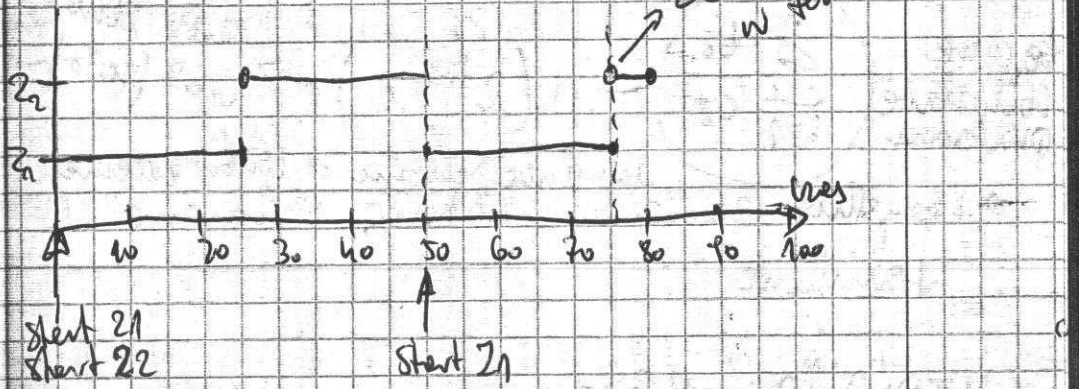
przykład 2.

$Z_1 = 25ms, 50ms$

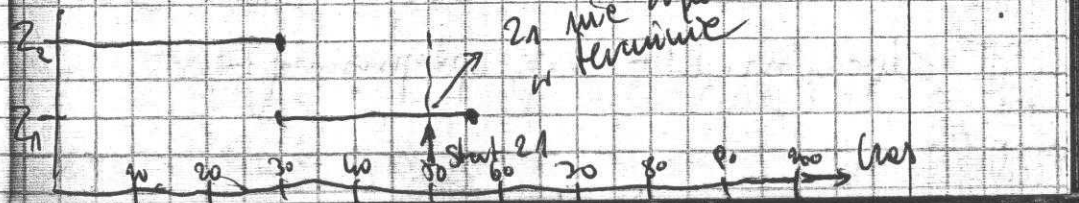
$Z_2 = 30ms, 75ms$

$$\text{obciążenie} = \frac{25}{50} + \frac{30}{75} = 0{,}5 + 0{,}4 = 0{,}9 = 80\%$$

(a) priorytet $Z_1 > Z_2$



(b) priorytet $Z_2 > Z_1$



Algorytm szeregowania RMS (Rate Monotonic Scheduling)
→ priorytet zadanie tym wyższy im wyższa częstotliwość powtarzania (im krótszy okres c_i)

Twierdzenie:
1. algorytm RMS jest optymalny w klasie algorytmów o stałych priorytetach
2. jeżeli spełniony jest warunek to →

$$\sum_{i=1}^{n} \frac{t_i}{c_i} \leq n\left(2^{\frac{1}{n}} - 1\right)$$

→ stałe liczba dolna granica dla dużych n: 0,69 (70%)

to czas obliczenia procesora ↓ stała liczba zależna od liczby zależnej

→ wszystkie zadania zostaną wykonane w terminie

MEDIANA – 88%

Algorytm szeregowania EDF – (Earliest Deadline First)

1. algorytm EDF jest optymalny

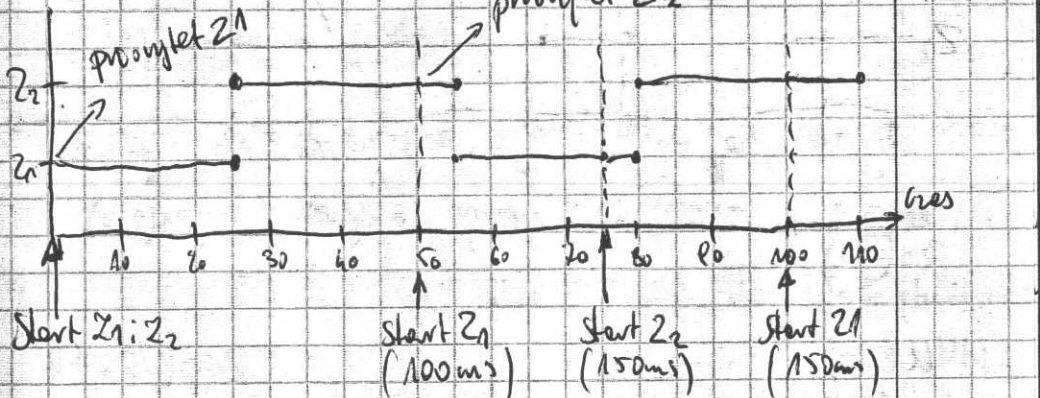
2. jeżeli

$$\sum_{i=1}^{n} \frac{t_i}{c_i} \leq 1$$

to wszystkie zadania zostaną wykonane w terminie

przykład:

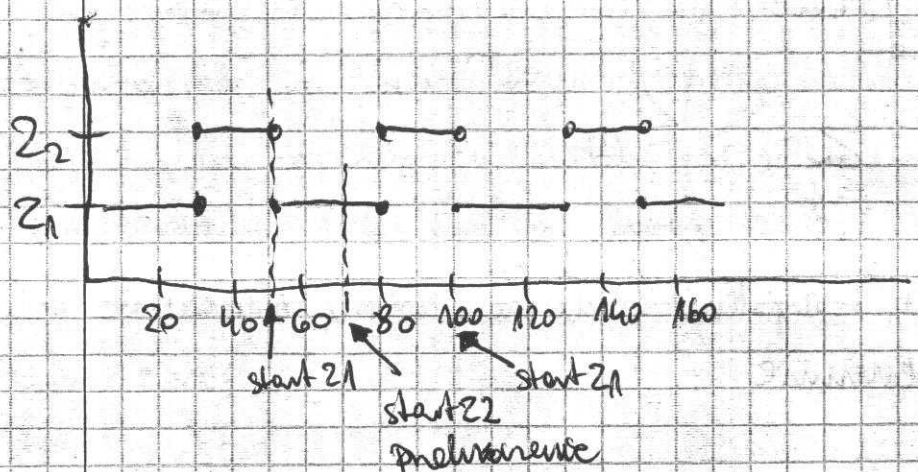
$Z_1 = (25\,ms,\ 50\,ms)$
$Z_2 = (30\,ms,\ 75\,ms)$



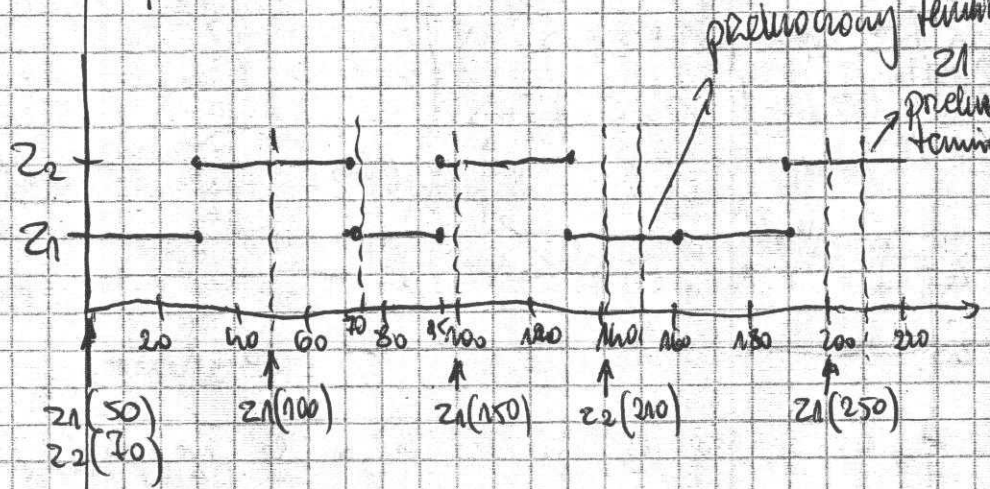
Przeciążenie systemu

$Z_1 = 30\,ms, 50\,ms$
$Z_2 = 35\,ms, 70\,ms$

obciążenie $\frac{30}{50} + \frac{35}{70} = 0,6 + 0,5 = 1,1$ (110%)



algorytm EDF



UKŁAD

2008/05/14

SYSTEM OPERACYJNY CZASU RZECZYWISTEGO

Wymagania:

- znany czas operacji systemowych
- kontrola czasu, w którym operacje potencjalnie

niezależnych
- kontrola poprawności nad szeregowaniem zadań
- wspomaganie współpracy zadań
- ograniczenie niedeterministycznej synchronizacji
- dostęp do przerwań i układów we/wy
- konfigurowalność

MODEL BUDOWY I DZIAŁANIA SYSTEMU

- program (zapis algorytmu w języku programowania)
- proces (ciąg)
- reprezentacje procesu w systemie
- wykonanie procesu

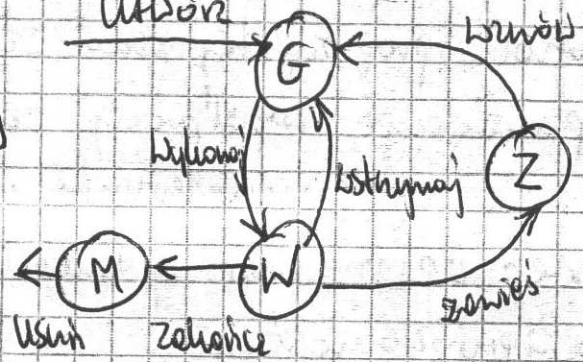
Proces - pojedyncze wykonanie programu
- zasoby (pamięć, czas procesora, urządzenie sterujące (kontroler
- wątek wykonania instrukcji
lub wątki - " - " - " -

Zombie

orphan - sierota

Stan procesu: Start utworz

- gotowy
- wykonywany
- zawieszony
- martwy



Zdarzenie:

- zewnętrzne
- czasowe
- wewnętrzne
- programowe

G gotowy
W wykonywany
Z zawieszony
M martwy

Lista rozkazów procesora

- dodaj
- przerwij
- pomnóż
- kopiuj do pamięci
- pisz wyjście
- czytaj wejście
- pisz do MMU

- czytaj z MMU
- przenumeruj układ pamięci

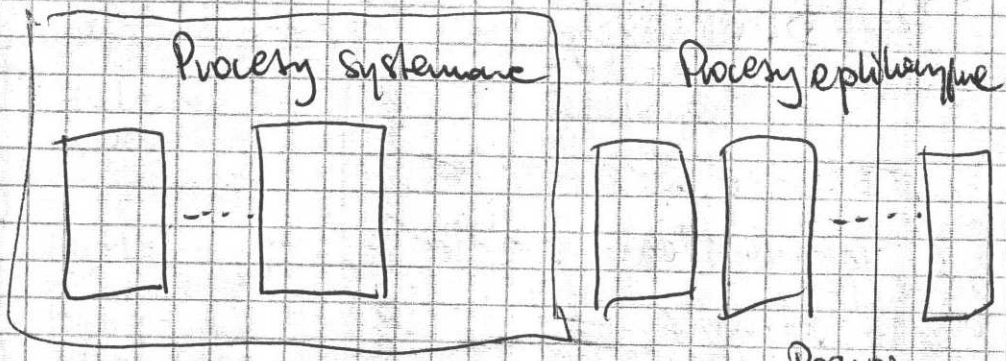
instrukcje uprzywilejowane

Tryby pracy procesora

- user (użytkownika)

MODEL SYSTEMU OPERACYJNEGO Z MIKROJĄDREM

- podział funkcji
- Architektura klient - serwer
- mechanizm dziedziczenia



POZIOM PROCESÓW

POZIOM EGZEKUTORA

System operacyjny

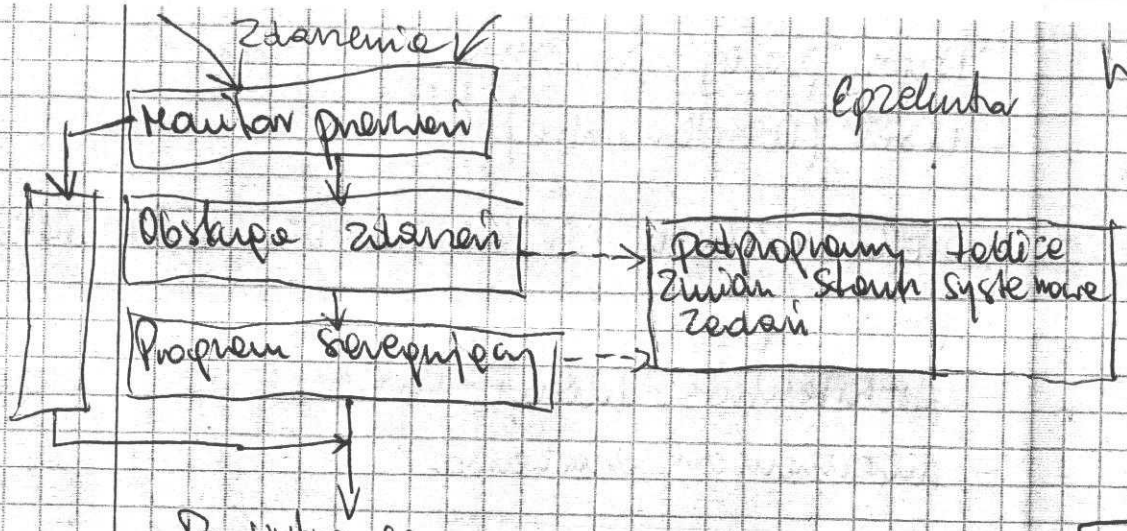
Wieloplatformowa egzekutywa wielozadaniowa

MODEL BUDOWY MIKROJĄDRA

- pojęcie sterowania
- obsługa zdarzeń
- szeregowanie i przełączanie kontekstu

WYKŁAD 31/05/2024

Egzelunka

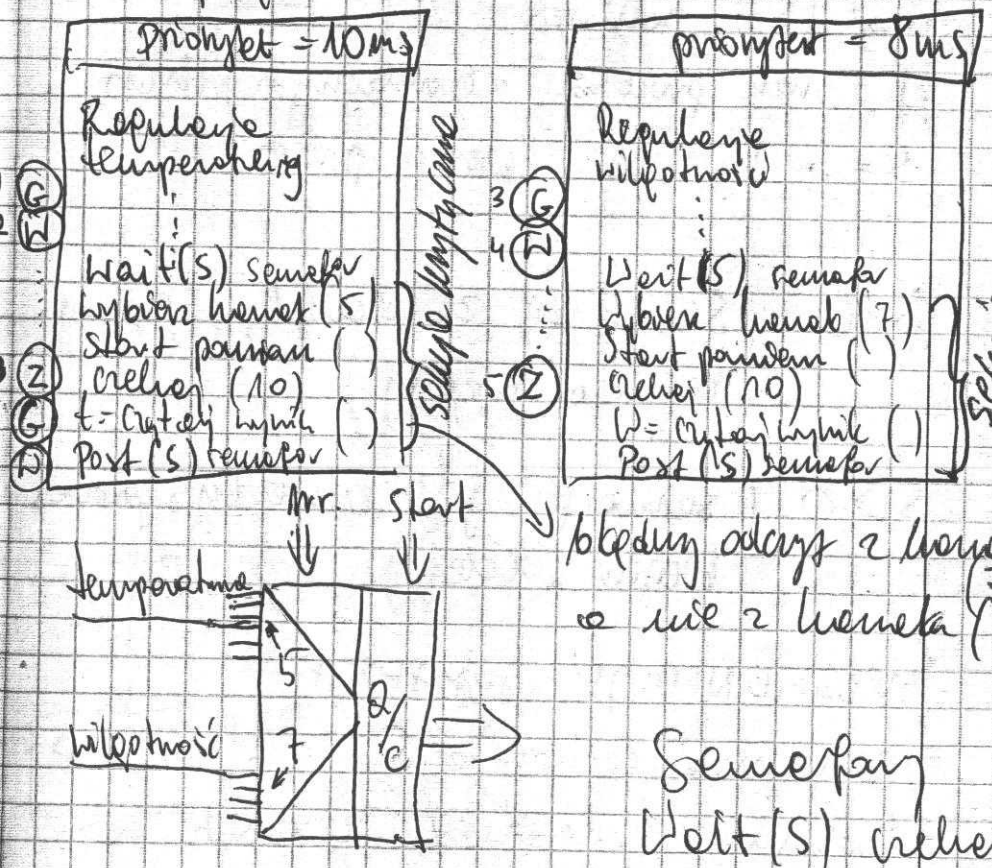


TABLICE SYSTEMOWE

1) Tablica stanu procesu
   - priorytet
   - stan bieżący
   - kontekst
   - identyfikatory
   - przydzielone obszary pamięci
   - inne przydzielone zasoby
2) Lista procesów gotowych
3) Listy zadań zablokowanych
4) Tablice stanu zasobów
5) Identyfikator procesu pid

Synchronizacja procesów

Konkurencja o wspólne zasoby
- obszar pamięci
- urządzenie (np. port szeregowy, przetwornik A/C)
- oprogramowanie



Semafor
Wait(S) czekaj
Post(S) podnieś

Semafor

Zmienna systemowa z niepodzielnymi operacjami

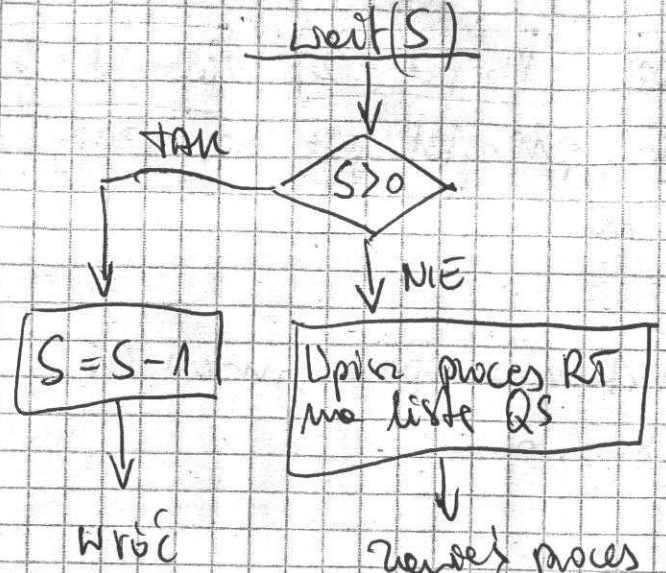
Wait(S): czekaj
if (S>0) then S = S-1 else zawieś wykonywany proces

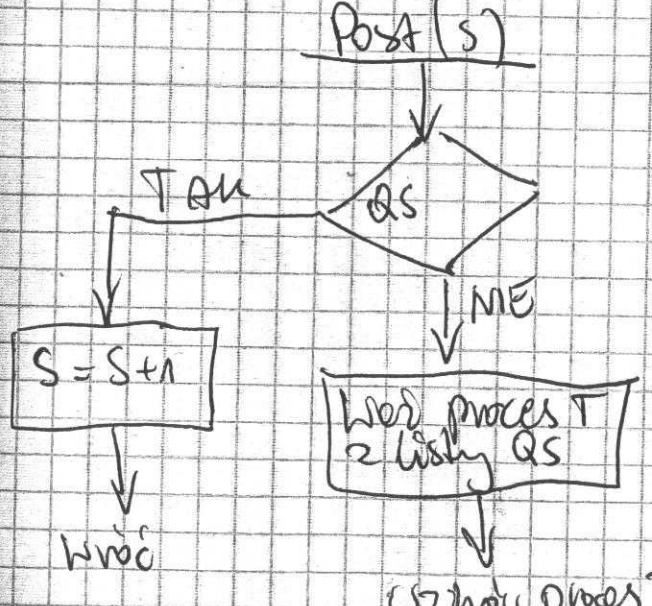
Post(S): podnieś
if (brak procesów zawieszonych) then S = S+1 else wznów proces

S=0 (semafor zamknięty, proces jest zawieszony)

S>0 (semafor jest otwarty, proces biegnie dalej)

IMPLEMENTACJA SEMAFORA
- zmienna całkowita S
- kolejka procesów zawieszonych QS





INWERSJA PRIORYTETÓW (Priority inversion)

Priorytet = 10	Priorytet = 2	Priorytet = 6

Najpilniejszy priorytet jest najmniejszy
czyli np. 10

Przegląd systemów operacyjnych RT
Platforma sprzętowa
VxWorks
QNX Neutrino
WINDOWS CE
LYNX OS
RT LINUX
eCOS