GENEROWANIE SYGNAŁÓW

na procesorach sygnałowych

Opracowanie: Grzegorz Szwoch

Politechnika Gdańska, Katedra Systemów Multimedialnych

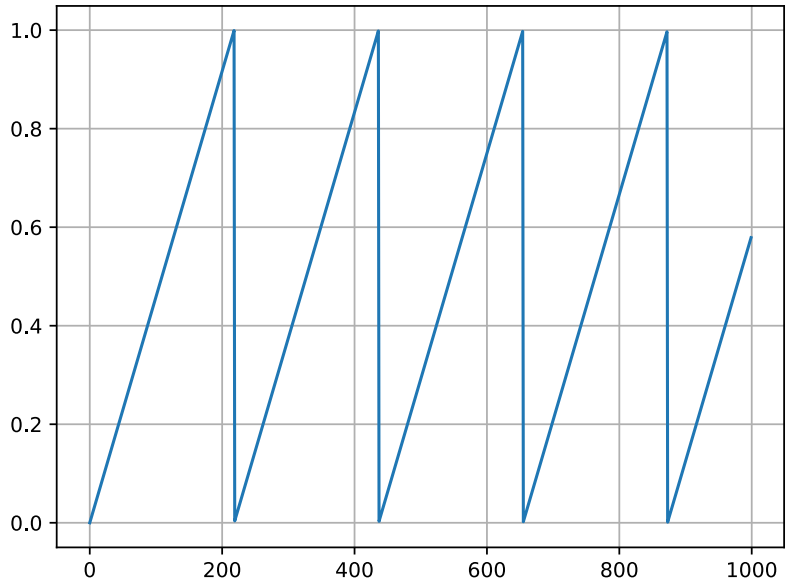
Wprowadzenie

- Procesory sygnałowe są zwykle stosowane do przetwarzania sygnałów podawanych na jego wejście.
- Możemy także wykorzystać procesor do generowania sygnałów – wejście nie jest wykorzystywane.
- Na tym wykładzie omówimy:
  - generowanie cyfrowych sygnałów harmonicznych,
  - generowanie sygnału sinusoidalnego różnymi metodami,
  - generowanie sygnału pseudoprzypadkowego,
  - generowanie dowolnych sygnałów z tablicy próbek,
  - interpolację próbek zapisanych w tablicy.
- Zastosowania: instrumenty muzyczne (syntezatory), generatory pomiarowe.

Sygnał fazowy

- Sygnał fazowy (*phasor*) lub akumulator fazowy to sygnał okresowy, w którym wartość zmienia się liniowo i okresowo od 0 do 1.
- Długość okresu: *T*, częstotliwość sygnału: *f* = 1 / *T*.
- Sygnał cyfrowy z częstotliwością próbkowania *fs*: wartość z próbki na próbkę zmienia się o krok *f* / *fs*.
- Implementacja w C w zapisie zmiennoprzecinkowym:

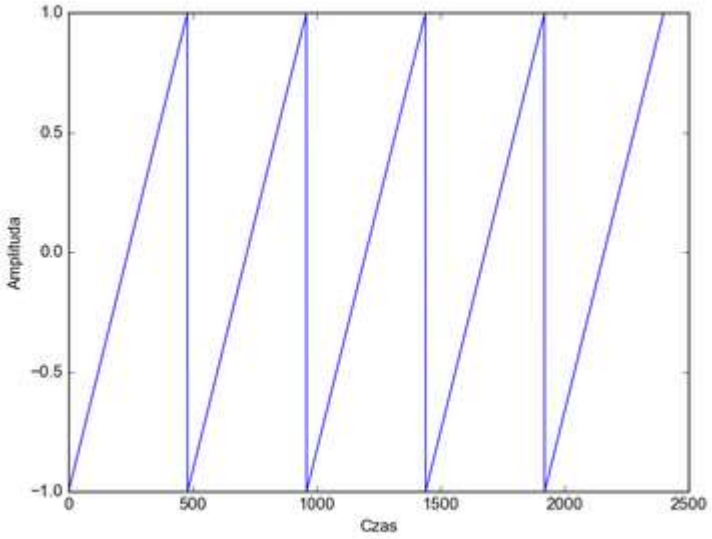
```
const float f = 440.f;          // częstotliwość sygnału fazowego
const float fs = 48000.f;       // częstotliwość próbkowania
const float krok = f / fs;      // krok zwiększania amplitudy
float y = 0.f;                  // wartość amplitudy
while (1) {
    y = y + krok;               // zwiększenie amplitudy o krok
    if (y >= 1.f)
        y -= 1.f;               // amplituda w zakresie (0, 1)
}
```

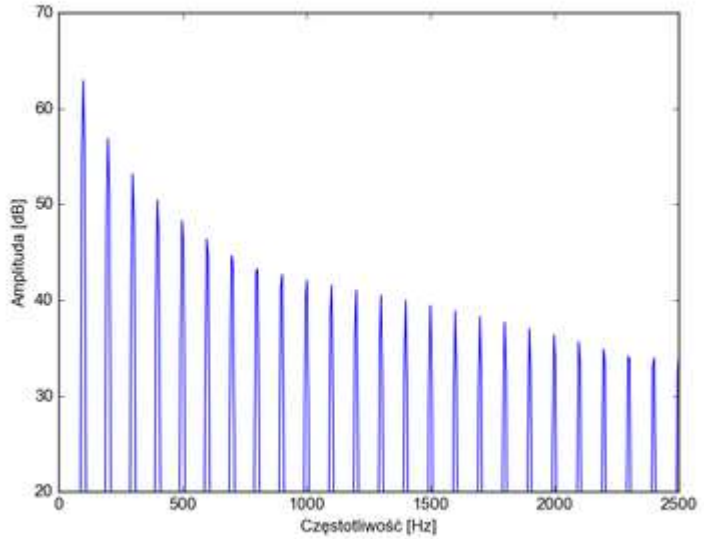


Fala piłokształtna

- Fala piłokształtna (*sawtooth wave*) zmienia się liniowo między minimum a maksimum, co okres.
- Jest to fala harmoniczna – widmo posiada składowe na całkowitych wielokrotnościach *f* (*f*, 2*f*, 3*f*, 4*f*, …).
- Możemy łatwo przekształcić sygnał fazowy w falę piłokształtną o amplitudzie *A*:

```
saw = 2 * A * (phasor - 0.5)
```





Fala piłokształtna na procesorze stałoprzecinkowym

- Używając zapisu stałoprzecinkowego możemy wygenerować falę piłokształtną wykorzystując przepełnienie zakresu liczby.
- Przy zapisie na 16 bitach mamy 2^{16} = 65536 wartości.
- Dla częstotliwości próbkowania 48000 Hz, wartość kroku dla częstotliwości fali *f*:

```
krok = f * 65536 / 48000 = f * 1,365333...
```

- Wartość trzeba zaokrąglić, np. dla *f* = 220 Hz: 300,37333… → 300
- Z uwagi na kwantyzację, nie możemy uzyskać dowolnej częstotliwości (tu: 219,73).

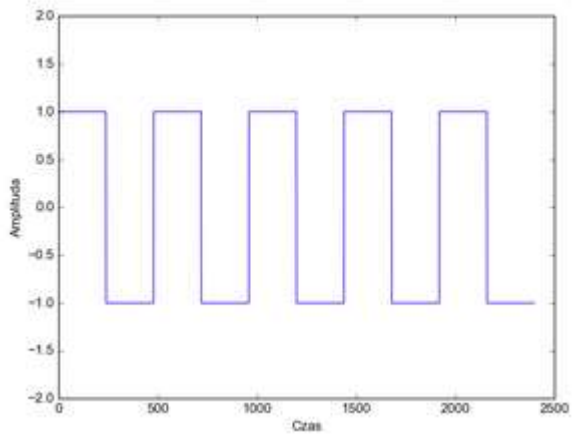
```
const short krok = 130;       // krok zwiększania amplitudy (obliczony dla danej częstotliwości fali)
short y = 0;                  // wartość amplitudy
while (1) {
    y = y + krok;             // zwiększenie amplitudy o krok, automatyczne zawinięcie
    // ... wykorzystanie obliczonej wartości fali piłokształtnej y
}
```

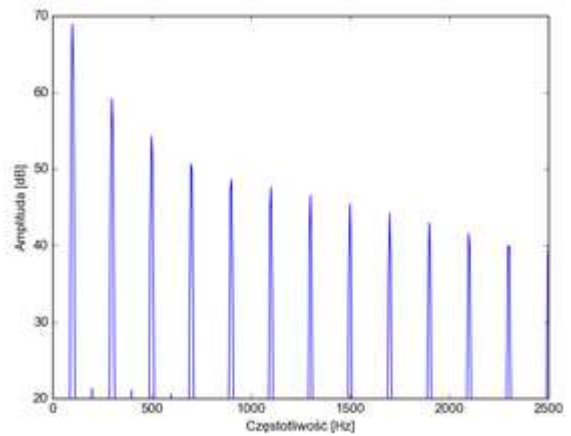
Fala prostokątna i impulsowa

- Fala prostokątna (*square wave*): dwustanowa, *+A* dla pierwszej połowy okresu i *–A* dla drugiej połowy okresu.
- Również fala harmoniczna, ale widmo ma tylko nieparzyste składowe (*f*, 3*f*, 5*f*, …).
- Fala impulsowa (*pulse wave*)*:* podobna do prostokątnej, ale jest asymetryczna. Długość części „dodatniej” jest równa szerokości impulsu (*pulse width*), od 0 do 1.
- Falę prostokątną i impulsową można uzyskać przez progowanie sygnału fazowego:

```
pulse =  A if phasor <  pulse_width
pulse = -A if phasor >= pulse_width
```

- Dla szerokości impulsu 0,5 mamy falę prostokątną.





Fala prostokątna i impulsowa

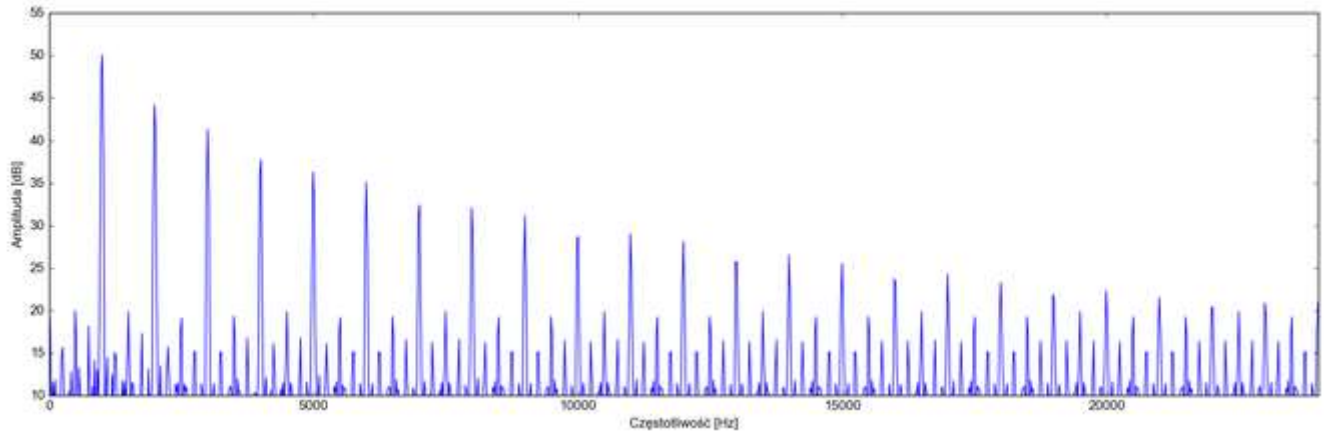
- Używając zapisu stałoprzecinkowego możemy wygenerować falę prostokątną i impulsową metodą progowania wygenerowanej fali piłokształtnej.
- Dla fali prostokątnej: wartość progu wynosi 0.

```
const short krok = 130;       // krok zwiększania amplitudy (obliczony dla danej częstotliwości fali)
short saw = 0;                // wartość sygnału piłokształtnego
short square;                 // wartość sygnału prostokątnego
while (1) {
    saw += krok;
    if (saw < 0) {
        square = 32767;
    } else {
        square = -32767;
    }
    // lub krócej: square = saw < 0 ? 32767 : -32767;
}
```

- Kod dla fali impulsowej pozostawiamy jako ćwiczenie.

Problem aliasingu

- Sygnały harmoniczne (analogowe) , takie jak prostokątny lub piłokształtny, mają nieskończone widmo, przekraczające częstotliwość Nyquista.
- Generując cyfrowo sygnały harmoniczne „z definicji” spowodujemy aliasing widma.
- Powstanie sygnał nieharmoniczny, zniekształcony.
- Aby uniknąć aliasingu, możemy np. generować fale z szeregu Fouriera.
- Istnieją specjalne metody redukcji aliasingu (MinBLEP, PolyBLEP)
  - zbyt złożone na ten wykład.



Problem aliasingu

- Przykład generowania fali piłokształtnej z szeregu Fouriera (wzór ilustracyjny, proszę się nie uczyć go na pamięć):

$$x(n) = \frac{A}{2} - \frac{A}{\pi}\sum_{k=1}^{N}(-1)^k \frac{\sin\left(2\pi k n f \,/\, fs\right)}{k}$$

- Sumujemy dopóki $k \cdot f$ nie przekracza częstotliwości Nyquista.
- Ta metoda wymaga dużo obliczeń.
- Podobne wzory istnieją dla innych fal harmonicznych.

Fala sinusoidalna

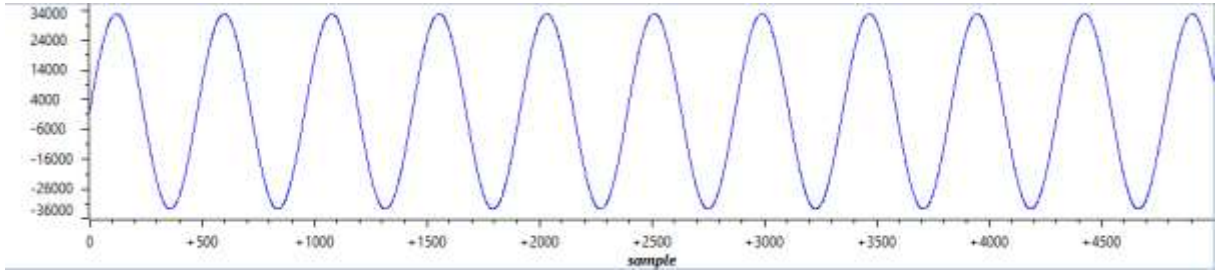
- Falę sinusoidalną można wygenerować z sygnału fazowego:

```
sine_wave = sin(2 * PI * phasor)
```

- Funkcja *sin* musi obliczać wartość sinusa dla kąta fazowego w radianach.
- Jeżeli nie mamy takiej funkcji, możemy ją aproksymować z szeregu Taylora:

$$\sin(x) \cong x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}$$

- Takie przybliżenie jest dokładne tylko dla „pierwszej ćwiartki” (0 do π/2).
- Sinus jest symetryczny, więc pozostałe trzy ćwiartki okresu można uzyskać poprzez symetrię z pierwszej ćwiartki.



Fala sinusoidalna

Na procesorze sygnałowym C5535 możemy wykorzystać funkcję *sine* z *DSPLIB*.

Podajemy do niej wartości wygenerowanej fali piłokształtnej.

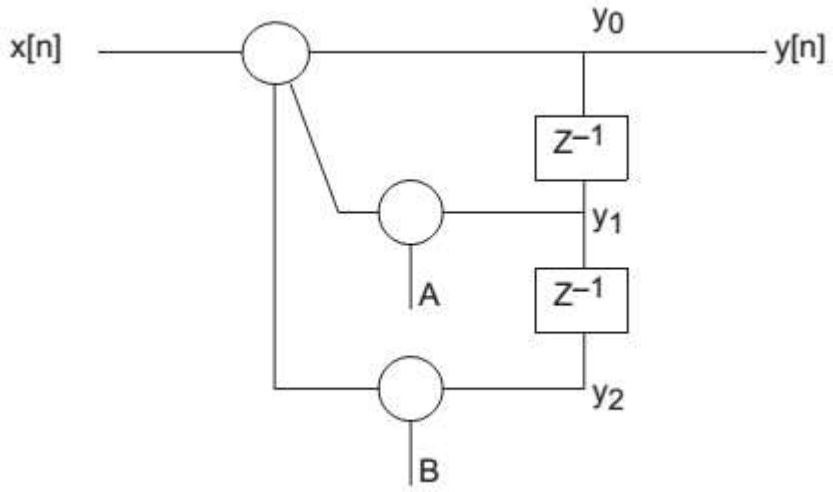
sine	*Sine*
Function	ushort oflag = sine (DATA *x, DATA *r, ushort nx)

```
const short krok = 130;       // krok zwiększania amplitudy (obliczony dla danej częstotliwości fali)
short y = 0;                  // wartość amplitudy
short bufor[1024];            // bufor na wartości sygnału piłokształtnego
short i;

for (i = 0; i < 1024; ++i) {
    y = y + krok;             // wartość fali piłokształtnej
    bufor[i] = y;             // zapisanie do bufora
}
sine(bufor, bufor, 1024);     // odczyt fazy z bufora, zapisanie sinusa do tego samego bufora
```

Sinus z układu IIR

- Alternatywna metoda generowania sinusa:
  stosujemy układ IIR drugiego rzędu na granicy stabilności.
- Pobudzamy filtr impulsem:
  $y(0) = -\sin(2\pi f/fs)$
- Później nie podajemy niczego na wejście.
- Układ wchodzi w oscylacje
  – generuje wartości sygnału sinus
  przy zerowym sygnale wejściowym.
- Implementacja na stałoprzecinkowym PS
  jest problematyczna
  (zbyt mała precyzja obliczeń)



$$y(n) = a \cdot y(n-1) - y(n-2)$$

$$a = 2\cos\left(\frac{2\pi\ f}{fs}\right)$$

Generowanie białego szumu

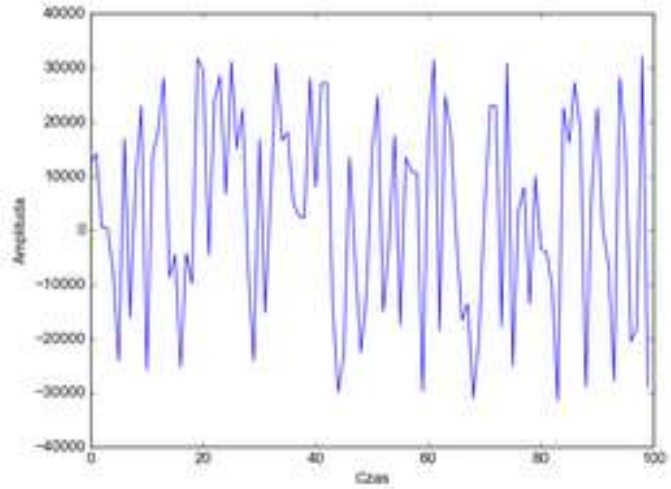
- Do generowania cyfrowego białego szumu (*white noise*) stosuje się generatory liczb pseudolosowych (*RNG – random numer generator*).
- Próbki są obliczane przez algorytm.
- Przykład prostego algorytmu generowania szumu:
  LCG – liniowy generator kongruentny:

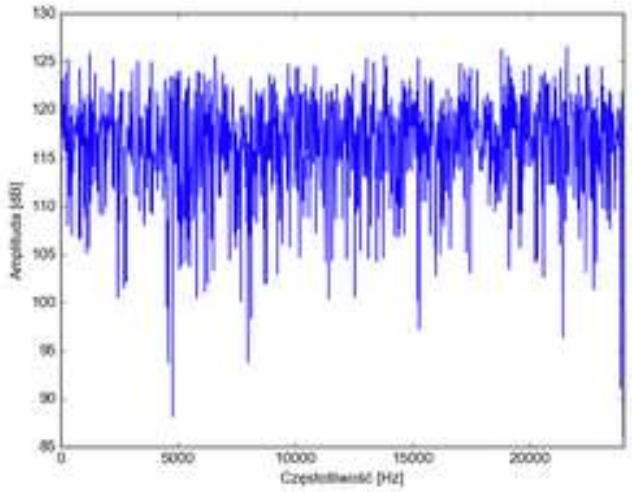
$$y(n) = \left[a \cdot y(n-1) + b\right] \bmod M$$

  mod – modulo, reszta z dzielenia przez *M*
- Przy zapisie liczb całkowitych na *N* bitach stosujemy $M = 2^N$.
- Do poważnych zastosowań (np. szyfrowanie danych) wymagane są dokładniejsze metody generowania liczb pseudolosowych (np. Mersenne Twister).

Generowanie białego szumu

- Wartość początkowa $y(0)$ to ziarno (*seed*).
- Podając to samo ziarno, dostaniemy zawsze taką samą sekwencję liczb.
- W praktyce: ustawiamy ziarno na zmienną, nieprzewidywalną liczbę, zazwyczaj aktualny czas z zegara systemowego.
- Przykład: $a = 2045$, $b = 0$, $M = 2^{20}$, $y(0) = 12345$.
  Postać czasowa i widmowa:





Generowanie szumu białego z DSPLIB

- Inicjalizacja generatora (wpisanie ziarna) – tylko raz na początku programu:

```
rand16init();
```

- Wypełnienie bufora *r* o długości *nr* próbkami:

rand16	*Random Number Generation Algorithm*
Function	ushort oflag= rand16 (DATA *r, ushort nr)

Jest to układ LCG z parametrami: $a = 31821$, $b = 13849$, $M = 65536$.

- Wypełnienie bufora wartościami białego szumu:

```
short bufor[1024];
rand16(bufor, 1024);
```

Sygnał z tablicy próbek

- Możemy mieć wartości jednego okresu dowolnej okresowej fali zapisane w tablicy (*wavetable*).
- Możemy wygenerować ciągłą falę odczytując cyklicznie wartości z tablicy, traktując ją jak bufor kołowy.
- Częstotliwość takiej fali jest stała i wynosi:
  częstotliwość próbkowania / długość tablicy
- Np. dla częstotliwości próbkowania 48 kHz i tablicy o długości 1024 dostaniemy falę o częstotliwości 48000 / 1024 = 46,875 Hz.
- W jaki uzyskać dowolną częstotliwość fali?
- Musimy odczytywać wartości z tablicy, przesuwając indeks odczytu o krok:

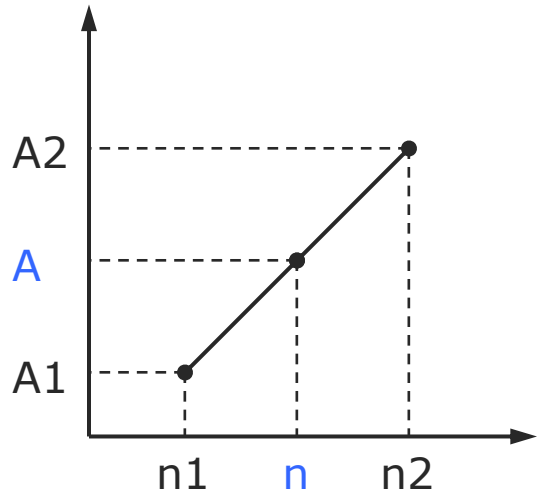
```
krok = długość_tablicy * częstotliwość_fali / częstotliwość_próbkowania
```

Sygnał z tablicy próbek

- Dla tablicy o długości 1024 próbek i częstotliwości próbkowania 48 kHz: jeżeli chcemy częstotliwość 220 Hz, to krok = 1024 × 220 / 48000 = 4,6933333…
- Nie mamy wartości dla indeksu 4,6933333…, mamy dla 4 i dla 5.
- Interpolacja liniowa: łączymy wartości tablicy linią prostą i znajdujemy wartość w żądanym miejscu.
- W naszym przykładzie (zapis zmiennoprzecinkowy):

```
y1 = tablica[4];
y2 = tablica[5];
alfa = 0.6933333;
y_interp = (1 – alfa) * y1 + alfa * y2;
```

- Jest to tylko przybliżenie. Im mniejsza długość tablicy, tym większe błędy interpolacji.



Interpolacja liniowa

Przykład interpolacji liniowej na stałoprzecinkowym procesorze sygnałowym.

```
ushort indeks_c = 45;             // część całkowita indeksu odczytu
ushort indeks_u = 39322;          // część ułamkowa indeksu odczytu (0,6) w UQ16
short y1 = tablica[indeks_c];     // ”poprzednia” wartość z tablicy wartości Q15
short y2 = tablica[indeks_c+1];   // ”następna” wartość z tablicy wartości Q15

// interpolacja liniowa, wynik w Q31
const long y = (32768L – (long)indeks_u) * y1 + (long)indeks_u * y2;

// zaokrąglenie Q31 i konwersja do Q15
short y_interp = _sround(y) >> 16;
```

Generowanie dźwięków muzycznych

- Możemy zapisać w pamięci nie tylko pojedyncze okresy fal, ale także dłuższe sygnały, np. dźwięki (instrumentów muzycznych lub inne).
- Dźwięki są odtwarzane poprzez odczyt próbek z pamięci, jednorazowo lub z zapętleniem wybranego fragmentu („loopy”).
- Taki sposób generowania nazywa się samplingiem, a instrument samplerem.
- Zmiana częstotliwości – tak jak dla tablicy fal, przez zmianę kroku odczytu.
- Jeżeli krok > 1, zwiększymy częstotliwość (wysokość) dźwięku, ale jednocześnie przyspieszymy dźwięk.
- Jeżeli krok < 1, zmniejszymy częstotliwość (wysokość) dźwięku, ale jednocześnie spowolnimy dźwięk.