Zadanie 3 – zajęcia 5, 6 i 7

Cel ćwiczenia

Celem zadania realizowanego w trakcie trzech ostatnich laboratoriów jest zaznajomienie studentów z funkcjami przerwania 10h BIOS, służącego do realizacji operacji graficznych. Ponadto w ramach ćwiczenia studenci zaznajomieni zostaną z podstawowymi aspektami programowania koprocesora arytmetyki zmiennoprzecinkowej.

Zadanie

W ramach zadania 3 każdy student zaimplementuje program realizujący prostą grafikę VGA. Oprócz operacji graficznych w przerwaniu 10h, program wykonywał będzie także pewne obliczenia zmiennoprzecinkowe. Szczegółowe omówienie architektury i listy instrukcji koprocesora arytmetyki zmiennoprzecinkowej dostępne w odnośniku.

Wszystkie niezbędne argumenty programu powinny być wczytane z wiersza poleceń. Program musi sprawdzić poprawność przekazanych argumentów oraz – jeśli to konieczne – dokonać ich konwersji na odpowiedni format wewnętrzny.

Na ocenę rozwiązania zadania składają się:

1. poprawność implementacji, w tym poprawne wykorzystanie rejestrów koprocesora arytmetyki zmiennoprzecinkowej

2. przejrzystość implementacji, w tym należyte skomentowanie poszczególnych partii instrukcji w programie, unikanie nadmiarowych/nieczytelnych instrukcji skoku oraz prawidłowy podział programu na procedury

3. terminowość realizacji

Rysowanie zbioru Mandelbrota

Program ma zadanie narysować fragment zbioru Mandelbrota. Bezpośrednio po uruchomieniu, program wczytuje z wiersza poleceń cztery liczbyzmiennoprzecinkowe: xmin, xmax, ymin i ymax. Zakładamy, że liczby wprowadzane są w formacie “klasycznym”, np. `1234.567`. Każdy inny format program powinien potraktować jako błąd. Po wczytaniu i konwersji liczb, program przechodzi w tryb graficzny i rozpoczyna rysowanie fragmentu zbioru Mandelbrota zawartego w kwadracie o narożnikach: lewy górny = `(xmin, ymin)` i prawy dolny = `(xmax, ymax)`. W tym celu program przechodzi w dwóch zagnieżdżonych pętlach po wszystkich pikselach na ekranie. Niech `N`, `M` będą współrzędnymi ekranowymi aktualnego piksela. Program konwertuje je na współrzędne w układzie `xmin-xmax/ymin-xmax`:

```
p = xmin + N*(xmax-xmin)/rozdzielczosc_ekranu_w_osi_x

q = ymin + M*(ymax-ymin)rozdzielczosc_ekranu_w_osi_y
```

Następnie program wykonuje poniższe operacje arytmetyczne (pseudokod) :

```
x = 0
y = 0
for(i = 0; I < 1000; i++)
      tmp = x*x - y*y + p
      y = 2*x*y + q
      x = tmp
      if(x*x + y*y > 4.0) break
if(i == 1000) wynik = 1
if(i < 1000) wynik = 0
```

Jeśli po wykonaniu powyższych operacji zachodzi warunek `wynik == 1` (co oznacza, że wykonano wszystkie 1000 iteracji pętli) to bieżący piksel `(N, M)` otrzymuje kolor biały. Jeśli natomiast po wykonaniu powyższych operacji zachodzi `wynik == 0` (co oznacza, że wykonano mniej niż 1000 iteracji ze względu na przerwanie pętli instrukcją break) to piksel otrzymuje kolor czarny.

Cały zbiór Mandelbrota otrzymamy po wykonaniu powyższych operacji dla wszystkich pikseli, przy parametrach: `xmin=-2 xmax=1 ymin=-1 i ymax=1`.