Laboratorium ARKO semestr letni 2009/2010

Projekt Intel

Obraz czarno-biały jest kodowany z użyciem jednego bitu na piksel. Najstarszy bit w bajcie odpowiada pikselowi położonemu najbardziej z lewej strony w obrazie. Linia obrazu jest wyrównywana w pliku do najbliższej wielokrotności 4 bajtów. W obrazie czarno-białym mamy takie zależności:
Liczba bajtów w linii, w których są piksele obrazu = (szerokość obrazu + 7)/8
Liczba bajtów w linii (w pliku) = (((szerokość obrazu + 7)/8 + 3)/4) * 4

Informacje o pamięci obrazu, oraz istotnych parametrach rysowania są zapamiętane w strukturze imgInfo:

```
struct {
     int w, h;   // szerokość i wysokość obrazu
     unsigned char* pImg; // bufor obrazu
     int cX, cY; // współrzędne aktualnego punktu rysowania
     int col;    // kolor rysowania (0 – czarny, 1 – biały)
} imgInfo;
```

Dodatkowo, do kopiowania prostokątnych obszarów obrazu przydadzą się struktury:

```
struct {
     int left, top; // współrzędne górnego, lewego rogu pr.
     int right, bottom;  // współrzędne dolnego, prawego
                    // rogu pr., ten punk należy do pr.
} Rect;

struct {
     int x, y; // współrzędne pozioma i pionowa punktu
} Point;
```

Proszę zaimplementować funkcję:

```
imgInfo* CopyRect(imgInfo* pImg, Rect* pSrc, Point* pDst);
```

gdzie

`pSrc` jest wskazaniem na współrzędne prostokąta źródłowego, który będzie kopiowany

`pDst` to współrzędne górnego, lewego rogu prostokąta docelowego

Docelowy prostokąt może być położony częściowo lub w całości poza obrazem – tę sytuację trzeba wykryć i odpowiednio oprogramować. Jednak największy problem będzie stanowić możliwość nakładania się na siebie prostokąta docelowego i źródłowego. Nie wszystkie takie sytuacje są szkodliwe (w sensie konieczności modyfikacji „naturalnej” kolejności kopiowania pikseli). Ale w niektórych sytuacjach, kolejność kopiowania pikseli powinna być zmieniona.

Oczywiście przykład podany w pliku graf_io.c może służyć jedynie jako bardzo niedoskonały wzór. W szczególności pomysł, żeby kopiować prostokątny obszar obrazu czytając i zapisując pojedyncze piksele jest beznadziejny.

Projektując swoją wersję funkcji CopyRect proszę zastanowić się, czy nie warto podzielić jej na część implementowaną w C oraz część implementowaną w asemblerze. Chodzi o to, żeby w asemblerze pisać tylko te części, które mają istotny wpływ na szybkość wykonywania funkcji.

Oczekuję, że na zajęciach konsultacyjnych (25 maja) przedstawicie mi Państwo dość szczegółowy scenariusz działania funkcji, wraz z podaniem, które fragmenty będą implementowane w C, a które w asemblerze.

Jeżeli macie Państwo jakiekolwiek pytania, nie zwlekajcie z ich zadawaniem.