Projekt Architektura Komputerów MIPS

Wiktor Ślęczka

2 grudnia 2013

Streszczenie

Program ma obliczyć odległość Hamminga pomiędzy dwoma 1-bitowymi bitmapami o rozdzielczości 64x64, uwzględniając możliwość przesunięcia o 7 pixeli w każdą stronę.

1 Program

1.1 Opis

Program liczy minimalną odległość Hamminga pomiędzy dwoma bitmapami. Zgodnie ze specyfikacją, bitmapa powinna być 1bitowa, oraz mieć rodzielczość 64x64. Ogległość należy policzyć dla każdego możliwego przesunięcia z zakresu <-7, 7> pixeli, zarówno poziomo jak i pionowo[1].

1.2 Wejście

Program wczytuje dwa pliki, 'obraz1.bmp' oraz 'obraz2.bmp', zawierające poprawne 1bitowe bitmapy o rozdzielczości 64x64.

1.3 Wyjście

Program powinien stworzyć dwa pliki:

- **hamming.txt** w którym znajduje się najmniejsza znaleziona wartość.
- **tablica.txt** w którym znajdują się wszystkie obliczone wartości.

lub wypisać odpowiedni komunikat błądu na konsolę.

2 Struktura programu

Program zawiera następujące procedury:

- **main** główna funkcja programu
- **counter** oblicza odległość Hamminga poprzez zawężanie zakresu do kolejno rzędu i półsłowa[2], a następnie zastosowanie w pętli operacji XOR dla wszystkich kombinacji i przesunięć bitów zawartych we wczytanym półsłowie. Operuje się tu na pojedyńczych bajtach.
- **popcount** algorytm zliczania jedynek, wykonywany w wersji dla 32 bitów. Przy pomocy odpowiednich przesunięć i masek każda jedynka traktowana jest jako swój własny licznik, a następnie dodawana do sąsiednich.

[1] Łącznie to 225 kombinacji
[2] Oryginalnie 'halfword'

- **minimal** znajduje najmniejszą wartość wśród obliczonych.
- **read** wczytuje plik o podanej nazwie.
- **write** zapisuje dany ciąg znaków do pliku.

3 Struktury danych

- **array** tablica z wynikami obliczeń
- **prints** tablica z wynikami obliczeń przekonwertowanymi na ASCII.
- **file<*>** miejsce na wczytywane pliki.
- **mask<*>** używane w programie maski binarne.

4 Algorytmy

4.1 Procedura counter

Procedura counter służy obliczeniu odległości Hamminga. Na początek wyszczególnia dwa wiersze, następnie dla każdego bitu poza ostatnim wczytuje kolejne dwa bity w odwrotnej kolejność niż w pamięci[3]. Następnie otrzymane półsłowa są przesuwane i obcinane do oktetów bitów, i za pomocą operacji XOR obliczana jest odległość Hamminga dla poszczególnych kombinacji oktetów. Powtarzane jest to odpowiednią ilość razy dla kolejnych wierszy, aby otrzymać przesunięcie w pionie. Obliczone wartości dodawane są do odpowiedniego pola w tabeli array.

Procedura ta wywołuje następujące podprocedury:

- **counter_vertical** iteruje po wierszach w drugim pliku. Dla danego wiersza z pliku 1 iteruje po wierszach w pliku drugim w granicach przesunięcia z zakresu <-7, 7> wierszy względem danego. Wiersze powyżej/poniżej końca wiersza są ignorowane. Dla każdej znalezionej pary wierszy uruchamiana jest procedura counter_row.
- **counter_row** jej zadaniem jest uruchomienie dla każdego bajtu podanych wierszy procedury counter_byte, za wyjątkiem ostatniego, który jest rozpatrywany oddzielnie za pomocą procedury counter_last_byte.
- **counter_byte** Wczytuje do pamięci dwa następujące bajty, a następnie, odpowiednio je przesuwając oraz stosując maskę dłudości bity, wykonuje operację XOR dla wszystkich możliwych poprawnych przesunięć w zakresie <-7, 7> bitów. Następnie zlicza jedynki w otrzymanym wyniku oraz dodaje wynik do uprzednio otrzymanego.
- **counter_last_byte** Funkcja analogiczna do counter_byte, ale biorąca pod uwagę fakt, że po pierwszym bajcie zaczyna się następny wiersz. Wczytuje tylko jeden bajt oraz obcina go do interesujących fragmentów przed wykonaniem operacji XOR dla wszystkich przesunięć.

4.2 Procedura popcount

Procedura popcount służy do zliczania jedynek. Polega ona na trzech prostych krokach. Najpierw należy przesunąć wartość rejestru, który zliczamy o 2^{N-1}, gdzie N to numer kroku, a następnie zastosować dla obu wartości AND z maską o sicie 2^{N-1} i dodać je do siebie. Należy powtarzać te kroki aż jeden z rejestrów w pierwszym kroku zostanie wyzerowany. W ten sposób każdy bit jest traktowany jako swój licznik, który następnie zlicza 2, 4, 8.. itd. bitów.

[3] Ponieważ bajty są przechowywane w postaci Little-Endian.

5 Testy

W katalogach test<n> znajdują się testy mające sprawdzić skrajne i klasyczne działanie programu. Są to odpowiednio:

1. Dwie białe plansze
2. Biała plansza z czarną kropką 4x4 w lewym górnym rogu i czysta biała plansza.
3. Biała i czarna plansza.
4. Biała plansza oraz biała plansza z czarną kropką pośrodku.
5. Litera A na białym tle przesunięta względem drugiego obrazka o [5, -2].

W katalogach znajdują się również pliki o rozszerzeniu .res, zawierające rozwiązania.