Wprowadzenie do programowania w języku C
grupa RKr, wtorek 16:15-18:00
lista nr 9 *(na pracownię 10.12.2018) (wersja 1)*

Zadanie 1. [15p na pracowni lub 10p po pracowni**]**

Wykorzystaj typ **MemDescriptor** z listy zadań 8. Napisz funkcję o sygnaturze „**int indexerZTiled**(**int** x, **int** y)", która utworzy 12-bitowy indeks przeplatając bity 6-bitowych współrzędnych x oraz y w następujący sposób:

indexerZTiled(x, y) → $y_5\ x_5\ y_4\ x_4\ y_3\ x_3\ y_2\ x_2\ y_1\ x_1\ y_0\ x_0$.

```
#define __mmask(b1, b2) ( (((1 << (b2+1)) -1) >> (b1)) << (b1) )
#define __liftbit(x, a, b)            /* assume (b >= a) */ \
  ( ( ((x) & (1 << (a))) << (b - a) ) /* expose lift-bit */ \
  | ((__mmask(a + 1, b) & (x)) >> 1)  /* create leap-set */ \
  | ( (~__mmask(a, b)) & (x) )        /* clean bit-space */ )
```

Przy pomocy __liftbit(x, a, b) zbuduj makro __chopbits(w), które pomiędzy każde dwa bity liczby w wstawi 0, __chopbits(w) = $0\ w_5\ 0\ w_4\ 0\ w_3\ 0\ w_2\ 0\ w_1\ 0\ w_0$. To pozwoli zaimplementować indekser oraz funkcję „**MemDescriptor layToZTiled**(**MemDescriptor** m)" kopiującą pamięć z deskryptora m konwertując układ pamięci na ZTiled z dowolnego układu w deskryptorze m; asymptotyczny czas indeksera to O(#bits).

Tak jak na poprzedniej liście stwórz szachownicę rozmiaru 64x64 znaków posiadającą pola rozmiaru 8x8. Zastosuj do niej layToZTiled(.) zmieniając układ pamięci na rekurencyjnie/hierarchicznie Z-kafelkowany. W szachownicy nadpisz jedną wybraną 64-znakową linię (jest ich 64) ignorując jej szyk/indekser:

ABCDEFGHIJKLMNOP############################abcdefghijklmnop

Po zapisaniu tej linii, stworzoną do tego celu funkcją „**void overwriteLine**(**MemDescriptor** m)", wypisz szachownicę na standardowym wyjściu używając szyku zgodnego w tej chwili z pamięciowym, czyli ZTiled.

Jeśli indeksy x oraz y składają się z liczby bitów, która jest potęgą dwójki, to można w elegancki sposób napisać funkcję, która wyliczy liniowy indeks w liczbie operacji proporcjonalnej do logarytmu z liczby bitów. W sprzęcie elektronicznym można zapleść ścieżki i taka operacja kosztuje jedną instrukcję niezależnie od liczby bitów, działając w asymptotycznym czasie O(1), nasza wersja będzie działała w czasie $O(\log_2(\#bits))$.

Załóż, że współrzędne mają 8 bitów i napisz funkcję o sygnaturze „**short chopbits8**(**short** x)" oraz funkcję o sygnaturze "**short zipbitsZ8**(**short** x, **short** y)", która podobnie jak indexerZTiled wyliczy:

zipbitsZ8(x, y) = $y_7\ x_7\ y_6\ x_6\ y_5\ x_5\ y_4\ x_4\ y_3\ x_3\ y_2\ x_2\ y_1\ x_1\ y_0\ x_0$.

Budżet na każdą z operacji bitowych (<< & |) wynosi $\log_2$(8 bitów) = 3 sztuki. Jeśli nie masz pomysłu na przebieg operacji bitowych, a brak satysfakcji Ci nie przeszkadza, zajrzyj do podpowiedzi na drugiej stronie :) Można zmienić układ Z na И (rosyjska litera 'i', zamiast niej do oznaczenia używa się czasem podobnego N). Zastanów się, co trzeba zrobić z bitami aby uzyskać układ o kształcie И?

Zadanie 2. [15p**]** Dostępne w serwisie SKOS.

post-overwrite 8x8 field:

```
A B E F # # # #
C D G H # # # #
I J M N # # # #
K L O P # # # #
# # # # a b e f
# # # # c d g h
# # # # i j m n
# # # # k l o p
```

bit-flow for chopbits8:

```
-----------------------------------------------------
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00  bits
-----------------------------------------------------
[] [] [] [] [] [] [] [] x7 x6 x5 x4 x3 x2 x1 x0 [basis]
[] [] [] [] x7 x6 x5 x4 x3 x2 x1 x0 [] [] [] [] [shift]
                        ## ## ## ##              trash
[] [] [] [] x7 x6 x5 x4 [] [] [] [] x3 x2 x1 x0 [clean] 0x0F0F
[] [] x7 x6 x5 x4 [] [] [] [] x3 x2 x1 x0 [] [] [shift]
            ## ##                    ## ##        trash
[] [] x7 x6 [] [] x5 x4 [] [] x3 x2 [] [] x1 x0 [clean] 0x3333
[] x7 x6 [] [] x5 x4 [] [] x3 x2 [] [] x1 x0 [] [shift]
      ##          ##          ##          ##     trash
[] x7 [] x6 [] x5 [] x4 [] x3 [] x2 [] x1 [] x0 [final] 0x5555
```

recursive/nested Z-Tiled pattern: