1.

在本題中，我先將 row base 的 kernel comp 的 function 做些微的更改，我將每個值都變成負的，如下第一張圖，其餘的部分沒有修改，並產生相應的 elf 檔，執行 Ripes 看 cache 的狀況。

我的 A 是 12*12 的方陣，只是將老師原本給的方陣繼續依相同規律做延伸而已，因此 B 的輸出為 10*10 方陣，如下第二張圖。

以下幾頁為三種要求的 cache 狀況。

```
b[row][col] += -a[row + i][col + j] * k[i][j];
b[row][col] += -a[row + i + 1][col + j] * k[i][j];
b[row][col] += -a[row + i][col + j + 1] * k[i][j];
b[row][col] += -a[row + i + 1][col + j + 1] * k[i][j];
```

```
-2, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, -2, 0, 1, 0, 0, 0, 0, 0, 0,
1, 0, -2, 0, 1, 0, 0, 0, 0, 0,
0, 1, 0, -2, 0, 1, 0, 0, 0, 0,
0, 0, 1, 0, -2, 0, 1, 0, 0, 0,
0, 0, 0, 1, 0, -2, 0, 1, 0, 0,
0, 0, 0, 0, 1, 0, -2, 0, 1, 0,
0, 0, 0, 0, 0, 1, 0, -2, 0, 1,
0, 0, 0, 0, 0, 0, 1, 0, -2, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, -2,
```

(1)

Index	V	D	LRU	Tag	Block 0	Block 1	Block 2	Block 3
0	1	1	0	0x00003fff	0x00000000	0x00000000	0x00000000	0x000ffff0
	0	0	3					
	0	0	3					
	0	0	3					
1	1	1	0	0x00003fff	0x00000000	0x00000000	0x00000000	0x00010818
	0	0	3					
	0	0	3					
	0	0	3					
2	1	1	0	0x00003fff	0x00000000	0x00000000	0x00000000	0x000107a8
	0	0	3					
	0	0	3					
	0	0	3					
3	1	1	0	0x00003fff	0x00000000	0x00000000	0x00000000	0x0001007c
	0	0	3					
	0	0	3					
	0	0	3					

Index	V	D	LRU	Tag	Block 0	Block 1	Block 2	Block 3
0	1	1	0	0x00003fff	0x00000000	0x00000000	0x00000000	0x000ffff0
	0	0	3					
	0	0	3					
	0	0	3					
1	1	1	0	0x00003fff	0x00000000	0x00000000	0x00000000	0x00010818
	0	0	3					
	0	0	3					
	0	0	3					
2	1	1	0	0x00003fff	0x00000000	0x00000000	0x00000000	0x000107a8
	0	0	3					
	0	0	3					
	0	0	3					
3	1	1	1	0x00003fff	0x00000000	0x00000000	0x00000000	0x0001007c
	1	1	0	0x00003ffe	0x00000000	0x00000000	0x00000000	0x00000000
	0	0	3					
	0	0	3					

觀察圖中 index = 3 的地方，原本已經有一個 way 被使用了，現在有第二筆 index 同為 3 的資料要塞進來，於是在下圖可以看到，該筆新的資料被放進來，且新放的資料 LRU 是 0，代表他是最近一次被使用的，而原本放在 index = 3 的另一筆資料的 LRU 就從 0 變成 1。

(2)

Index	V	D	LRU	Tag	Block 0	Block 1	Block 2	Block 3
0	1	1	0	0x00003fff	0x00000000	0x0000000a	0x0000000a	0x00011a88
	1	0	1	0x00000470	0x00000000	0x00000000	0x00000000	0xffffffff
	1	0	3	0x0000046e	0x00000000	0x00000000	0xffffffff	0x00000000
	1	0	2	0x0000046f	0x00000000	0xffffffff	0x00000000	0x00000000
1	1	1	0	0x00003fff	0x00000002	0x00000002	0x0000000a	0x0000000a
	1	0	1	0x00000470	0x00000000	0x00000002	0x00000000	0x00000000
	1	0	3	0x0000046e	0x00000002	0x00000000	0xffffffff	0x00000000
	1	0	2	0x0000046f	0x00000000	0x00000000	0x00000000	0x00000000
2	1	0	2	0x0000046f	0xffffffff	0x00000000	0x00000002	0x00000000
	1	0	3	0x0000046e	0x00000000	0x00000000	0x00000000	0x00000000
	1	0	0	0x00003fff	0x00000000	0x00000000	0x00100000	0x00010800
	1	1	1	0x00003ffe	0x00000000	0x00000000	0x00000000	0x00010830
3	1	0	3	0x0000046d	0x00000000	0x00000000	0x00000000	0x00000000
	1	1	0	0x00003ffe	0x00000000	0x00000000	0x00000000	0x000ffff0
	1	0	2	0x0000046e	0x00000000	0xffffffff	0x00000000	0x00000002
	1	0	1	0x0000046f	0x00000000	0x00000000	0x00000000	0x00000000

Index	V	D	LRU	Tag	Block 0	Block 1	Block 2	Block 3
0	1	1	0	0x00003fff	0x00000000	0x0000000a	0x0000000a	0x00011a88
	1	0	1	0x00000470	0x00000000	0x00000000	0x00000000	0xffffffff
	1	0	3	0x0000046e	0x00000000	0x00000000	0xffffffff	0x00000000
	1	0	2	0x0000046f	0x00000000	0xffffffff	0x00000000	0x00000000
1	1	1	0	0x00003fff	0x00000002	0x00000002	0x0000000a	0x0000000a
	1	0	1	0x00000470	0x00000000	0x00000002	0x00000000	0x00000000
	1	0	3	0x0000046e	0x00000002	0x00000000	0xffffffff	0x00000000
	1	0	2	0x0000046f	0x00000000	0x00000000	0x00000000	0x00000000
2	1	0	2	0x0000046f	0xffffffff	0x00000000	0x00000002	0x00000000
	1	0	3	0x0000046e	0x00000000	0x00000000	0x00000000	0x00000000
	1	0	0	0x00003fff	0x00000000	0x00000000	0x00100000	0x00010800
	1	1	1	0x00003ffe	0x00000000	0x00000000	0x00000000	0x00010830
3	1	0	0	0x00003fff	0x00000000	0x00000000	0x00000000	0x0001007c
	1	1	1	0x00003ffe	0x00000000	0x00000000	0x00000000	0x000ffff0
	1	0	3	0x0000046e	0x00000000	0xffffffff	0x00000000	0x00000002
	1	0	2	0x0000046f	0x00000000	0x00000000	0x00000000	0x00000000

觀察 index = 3 的 4 個 way，原本 4 個 way 都已經塞滿了，現在又有一筆 index = 3 的資料要放進來，所以我必須將裡面 LRU 最高(LRU = 3)的那一筆資料換成新的，也就是更新醉酒之前使用的那一筆。更新之後，新進來的那筆資料 LRU = 0，而舊有的留下來的那三筆資料的 LRU 都+1。

(3)

Index	V	D	LRU	Tag	Block 0	Block 1	Block 2	Block 3
0	1	1	0	0x00003fff	0x00000000	0x00011a88	0x00010a78	0x00010838
	1	1	1	0x0000046a	0x00000000	0x00000000	0x00000002	0x00000000
	1	0	2	0x00000421	0x00000001	0x00000001	0x00000001	0x00000001
	0	0	3					
1	1	1	0	0x00003fff	0x00000001	0x00000000	0x00000000	0x00000000
	0	0	3					
	0	0	3					
	0	0	3					
2	1	1	1	0x00003fff	0x00000000	0x00000000	0x00100000	0x000107c4
	1	0	0	0x00000421	0x00000001	0x00000001	0x00000001	0x00000002
	0	0	3					
	0	0	3					
3	1	1	3	0x00003fff	0x00000000	0x00000000	0x00000000	0x0001007c
	1	1	2	0x00003ffe	0x00000000	0x00000000	0x00000000	0x00000000
	1	0	1	0x00000420	0x0000000a	0x0000203a	0x00000002	0x00000001
	1	0	0	0x00000429	0x00000001	0x00000002	0x00000001	0xffffffff

Index	V	D	LRU	Tag	Block 0	Block 1	Block 2	Block 3
0	1	1	0	0x00003fff	0x00000000	0x00011a88	0x00010a78	0x00010838
	1	1	1	0x0000046a	0x00000000	0x00000000	0x00000002	0x00000000
	1	0	2	0x00000421	0x00000001	0x00000001	0x00000001	0x00000001
	0	0	3					
1	1	1	0	0x00003fff	0x00000001	0x00000000	0x00000000	0x00000000
	0	0	3					
	0	0	3					
	0	0	3					
2	1	1	1	0x00003fff	0x00000000	0x00000000	0x00100000	0x000107c4
	1	0	0	0x00000421	0x00000001	0x00000001	0x00000001	0x00000002
	0	0	3					
	0	0	3					
3	1	0	0	0x00000421	0x00000001	0x00000001	0x00000001	0x00000001
	1	1	3	0x00003ffe	0x00000000	0x00000000	0x00000000	0x00000000
	1	0	2	0x00000420	0x0000000a	0x0000203a	0x00000002	0x00000001
	1	0	1	0x00000429	0x00000001	0x00000002	0x00000001	0xffffffff

觀察 index = 3 的第一個 way，原本 LRU=3，且 dirty bit = 1，代表該資料和 memory 的資料是 inconsistent 的，下個瞬間更新該筆資料，便將正確的值寫回 memory，所以上下兩張圖可以觀察到更新前後 dirty bit 由 1 變成 0，LRU 由 3 變成 0。

2.

在本題中，我先將 code 做修改，改成 block base 的形式，也就是多兩層迴圈去控制座標，迴圈如下第一張圖，執行結果如下第二張圖，與第一大題是相同的，證明我寫的 block base 結果正確。(有一起繳交 code 以供驗證)

```
for (int row = 0; row < Ar - 2; row += 2)
{
    for (int col = 0; col < Ac - 2; col += 2)
    {
        for (int r_offset = 0; r_offset <= 1; r_offset++)
        {
            for (int c_offset = 0; c_offset <= 1; c_offset++)
            {
                printf("r = %d, c = %d\n", row + r_offset, col + c_offset);
                for (int i = 0; i <= 1; i++)
                {
                    for (int j = 0; j <= 1; j++)
                    {
                        // printf("i = %d, j = %d\n", i, j);
                        // printf("row + r_offset + i = %d, col
                        b[row + r_offset][col + c_offset] += -a[row + r_offset + i][col + c_offset + j] * k[i][j];
                        b[row + r_offset][col + c_offset] += -a[row + r_offset + i + 1][col + c_offset + j] * k[i][j
                        b[row + r_offset][col + c_offset] += -a[row + r_offset + i][col + c_offset + j + 1] * k[i][j
                        b[row + r_offset][col + c_offset] += -a[row + r_offset + i + 1][col + c_offset + j + 1] * k[
                    }
                }
            }
        }
    }
}
```

```
-2, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, -2, 0, 1, 0, 0, 0, 0, 0, 0,
1, 0, -2, 0, 1, 0, 0, 0, 0, 0,
0, 1, 0, -2, 0, 1, 0, 0, 0, 0,
0, 0, 1, 0, -2, 0, 1, 0, 0, 0,
0, 0, 0, 1, 0, -2, 0, 1, 0, 0,
0, 0, 0, 0, 1, 0, -2, 0, 1, 0,
0, 0, 0, 0, 0, 1, 0, -2, 0, 1,
0, 0, 0, 0, 0, 0, 1, 0, -2, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, -2,
```

之後，我也將這個 block case 的版本的 elf 檔丟進 Ripes 觀察 cache 的情況，可以發現整個執行完成之後 hit rate 有些微的上升，hit rate 從原本 row base 的 99.51%(下面第一張圖)上升到現在 block base 的 99.64%，但其實不管是哪一個，其 hit rate 都相當接近 100%，可見 4-way 對於 hit rate 的提升有相當顯著之影響。