pro3

替换策略

LRU替换策略：

- 将被访问的数据放在头部
- 将最长时间未被访问的数据置换

```
44,46d43
<     if data[2] == 0:
<         return
<     data = data[0]
63c60,61
<     index = list(cache.keys())[0]
---
>     from random import choice
>     index = choice(list(cache.keys()))
86,89c84
<         cache[index] = [block, 0, 0]
<     else:
<         value = cache.pop(index)
<         cache[index] = value
---
>         cache[index] = block
91c86
<     x = cache[index][0][offset]
---
>     x = cache[index][offset]
106,109c101
<         cache[index] = [block, 0, 0]
<     else:
<         value = cache.pop(index)
<         cache[index] = value
---
>         cache[index] = block
111,112c103
<     cache[index][0][offset] = data
<     cache[index][2] = 1
---
>     cache[index][offset] = data
```

结果：

Random9.61 -> FIFO9.64 -> LRU9.74

```
Pass Correctness Check!
总共访存量为337.5MiB，在这过程中与主存交互字节数7.187MiB，如果不使用cache，共需与主存交互2.637GiB字节数据！
总共访问cache 11059200次，总共访问主存29436次，假设主存的访问时间为cache的10倍，则整体访存效率提高了9.74倍！
```

写回修改

增加dirty位。

结果：

无dirty9.74 -> 有dirty9.75

```
Pass Correctness Check!
总共访存量为337.5MiB，在这过程中与主存交互字节数6.968MiB，如果不使用cache，共需与主存交互2.637GiB字节数据！
总共访问cache 11059200次，总共访问主存28542次，假设主存的访问时间为cache的10倍，则整体访存效率提高了9.75倍！
```