1. Cache mapping example
    a. Continued from the previous notes
    b. Current cache: 8-byte 2-way SA cache with line size of 2, and a 4-bit address
    c. Address format for this cache below

Tag	**Set**	**Offset**		**Address Bits**
4 – 2 – 0 = 2 bits	$\log_2 S = \log_2 2 = 1$ bit	$\log_2 LS = \log_2 2 = 1$ bit	=	4 bits

    d. Use same addresses and data from previous two examples

Address	**Data**
0110	0x1B
0111	0x59
1000	0xFE
1001	0x3D
1110	0x0C
1111	0x3A
1010	0x25
1011	0x98

    e. Load addresses into cache into same fashion
        i. One set bit this time

Memory Access	**Tag**	**Set**	**Offset**	**Access Type**	**Data**	**Hit or Miss**
0110	01	1	0	Read		Miss
1000	10	0	0	Read		Miss
1110	11	1	0	Read		Miss
1010	10	1	0	Read		Miss
1110	11	1	0	Write	DE	Hit

    f. Place values from accesses into table below
        i. How do we know which line in set 1 to evict?

Set	**Line in Cache**	**Tag**	**Byte 0**	**Byte 1**	**Dirty**
0	0	10	FE	3D	0
	1				
1	2	~~01~~	~~1B~~	~~59~~	~~0~~
		10	25	98	0
	3	11	~~0C~~ DE	3A	~~0~~ 1

2. Cache replacement algorithms
    a. How do we evict lines from a given set for non-DM caches?
        i. Why do we not need a replacement policy for DM caches?
    b. Least recently used (LRU)
        i. Whichever line that has gone the longest without being touched is evicted
        ii. 2-way cache example: when we touch a line, set its *use* bit and clear the other line's bit

- c. First in, first out (FIFO)
    - i. Evict whichever line has been in the cache the longest
    - ii. Use a queue to implement
- d. Least frequently used (LFU)
    - i. Add a counter to each line
    - ii. Evict whichever line has the fewest accesses
- e. Random
    - i. Works almost as well as any of the others

3. Cache write policies
    - a. If we change values that reside in main memory, we make changes in cache first
    - b. What happens when we need to evict a line?
    - c. Write-through
        - i. All write operations are made to main memory as well as cache
        - ii. Every time a value is changed, must update RAM as well
        - iii. Requires writing to RAM extremely often
        - iv. Write-through does *not* use a dirty bit
    - d. Write-back
        - i. Only update RAM if evicted line's dirty bit is set
        - ii. Dirty bit set if CPU had to update value of line in cache
        - iii. Does not require writing to RAM nearly as often
            1. If we don't write to a line, then evict it, don't need to update main memory
        - iv. However, RAM value doesn't always match the updated cache value anymore
            1. Other devices that need the most up-to-date value must go through cache now