Benny Netzer
CSCI 454 - Computer Architecture
4/21/2022
Homework #5

1. **Warm-up**

BLOCK	**TAG**	**WORD 0**	**WORD 1**	**HIT/MISS LOG**
0a	2	Mem[32]	Mem[33]	**00 000 1**: B=0, W=1, T=0 **MISS**
0b	0	Mem[0]	Mem[1]	**01 001 0:** B=1, W=0, T=1 **MISS**
1a	2	Mem[34]	Mem[35]	**00 001 0**: B=1, W=0, T=0 **MISS**
1b	0	Mem[2]	Mem[3]	**00 001 1**: B=1, W=1, T=0 **HIT**
2a	0	Mem[4]	Mem[5]	**00 010 0**: B=2, W=0, T=0 **MISS**
2b	1	Mem[20]	Mem[21]	**01 010 0**: B=2, W=0, T=1 **MISS**
3a				**00 010 1**: B=2, W=1, T=0 **HIT**
3b				**01 010 1**: B=2, W=1, T=1 **HIT**
				10 000 1: B=0, W=1, T=2 **MISS**
				10 001 0: B=1, W=0, T=2 **MISS**
				00 000 1: B=0, W=1, T=0 **HIT**
				00 010 0: B=2, W=0, T=0 **HIT**
			Hit Rate: **41.67 %** (5 hits, 12 addresses)	

2. **Coding**
   I had already coded functionality for a set-associative cache in the last homework (oops). The only thing to do was to make the LRU replacement scheme work (before it worked entirely at random). In order to accomplish this, the most recent block accessed was inserted at the beginning of the set, and the last accessed was allowed to be overwritten if space required.

```
def load(self, address):
    set = self.find_set(address)
    tag = self.find_tag(address)
    for i in range(len(self.metaCache[set])):
        self.metaCache[set][i] = self.metaCache[set][i-1]
        self.cache[set][i] = self.cache[set][i-1]
    self.metaCache[set][0] = tag
    for i in range(0, self.blockSize):
        self.cache[set][0][i] = address + i
```

After this change, the test_q stream no longer worked. Then it became clear that find(self, address) also had to be modified.

```
def find(self, address):
    found = False
    set = self.find_set(address)
    tag = self.find_tag(address)
    for i in range(0,len(self.metaCache[set])):
        if self.metaCache[set][i] == tag:
            found = True
            block = self.cache[set][i]
            for j in range(0, i):
                self.metaCache[set][j+1] =
self.metaCache[set][j]
                self.cache[set][j+1] = self.cache[set][j]
            self.metaCache[set][0] = tag
            self.cache[set][0] = block
            break
    if found:
        self.hit = self.hit + 1.0
    return found
```