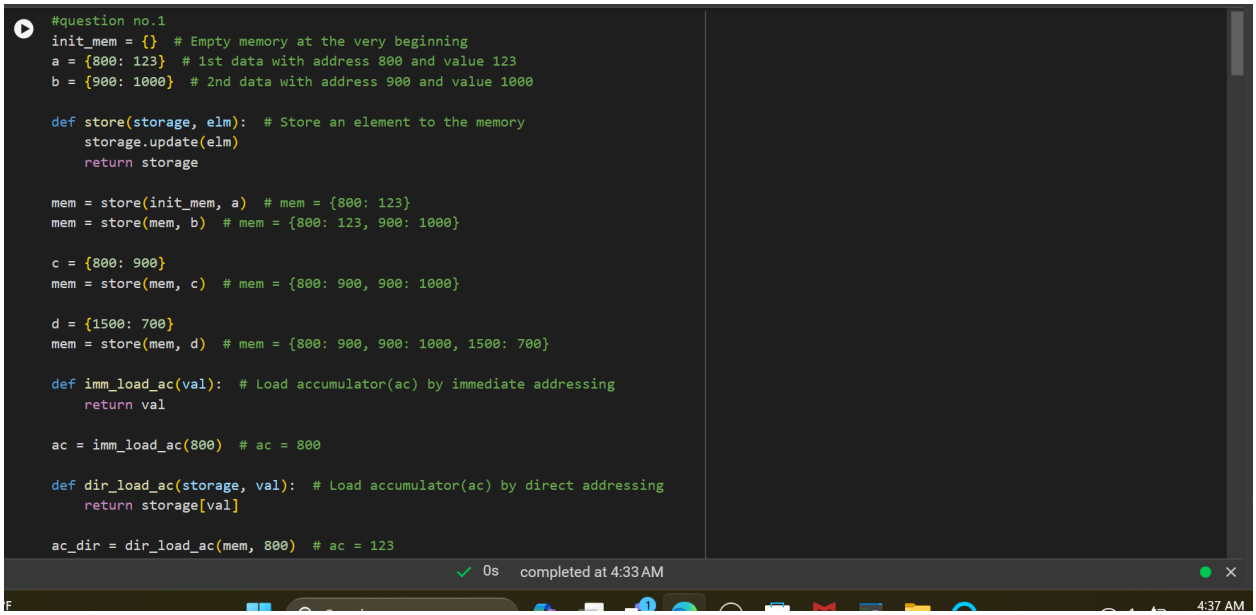
#question no.1



```
#question no.1
init_mem = {}  # Empty memory at the very beginning
a = {800: 123}  # 1st data with address 800 and value 123
b = {900: 1000}  # 2nd data with address 900 and value 1000

def store(storage, elm):  # Store an element to the memory
    storage.update(elm)
    return storage

mem = store(init_mem, a)  # mem = {800: 123}
mem = store(mem, b)  # mem = {800: 123, 900: 1000}

c = {800: 900}
mem = store(mem, c)  # mem = {800: 900, 900: 1000}

d = {1500: 700}
mem = store(mem, d)  # mem = {800: 900, 900: 1000, 1500: 700}

def imm_load_ac(val):  # Load accumulator(ac) by immediate addressing
    return val

ac = imm_load_ac(800)  # ac = 800

def dir_load_ac(storage, val):  # Load accumulator(ac) by direct addressing
    return storage[val]

ac_dir = dir_load_ac(mem, 800)  # ac = 123
```

0s completed at 4:33 AM

4:37 AM

```
ac_dir = dir_load_ac(mem, 800)  # ac = 123

def indir_load_ac(storage, val):  # Load accumulator(ac) by indirect addressing
    return storage.get(storage.get(val, 0), 0)

ac_indir = indir_load_ac(mem, 800)  # ac = 1000

def idx_load_ac(storage, idx, val):  # Load accumulator(ac) by Indexed addressing
    return storage.get(idx + val, 0)

idxreg = 700
ac_idx = idx_load_ac(mem, idxreg, 800)  # ac = 700

# Output
print("Memory:", mem)
print("Accumulator (Immediate):", ac)
print("Accumulator (Direct):", ac_dir)
print("Accumulator (Indirect):", ac_indir)
print("Accumulator (Indexed):", ac_idx)
```

```
Memory: {800: 900, 900: 1000, 1500: 700}
Accumulator (Immediate): 800
Accumulator (Direct): 900
Accumulator (Indirect): 1000
Accumulator (Indexed): 700
```

#question no.2

```
#question no.2
init_mem = {}

def store(storage, elm):
    storage.update(elm)
    return storage

mem = store(init_mem, {"00000110101000": [0, 1, 2, 3, 4, 5, 6, 7]})

b = {"00001110101000": [10, 11, 12, 13, 14, 15, 16, 17]}
mem = store(mem, b)

cache = {
    "0000": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "0001": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "0010": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "0011": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "0100": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "0101": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "0110": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "0111": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "1000": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "1001": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "1010": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "1011": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "1100": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
    "1101": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
```

0s completed at 4:33 AM

Search 4:38 AM 12/8/2023

```
def dir_map_cache(cache, adr, storage):
    block_label = adr[4:8]
    tag = adr[:7]
    if block_label not in cache:
        cache[block_label] = [tag, storage.get(adr, [0, 0, 0, 0, 0, 0, 0, 0]), 1]
    else:
        cache[block_label] = [tag, storage.get(adr, [0, 0, 0, 0, 0, 0, 0, 0]), 1]
    return cache

adr1 = "00000110101010"  # hex address: 1AA
cache = dir_map_cache(cache, adr1, mem)

adr2 = "00001110101010"  # hex address: 3AA
cache = dir_map_cache(cache, adr2, mem)

c = {"00001110111000": [20, 21, 22, 23, 24, 25, 26, 27]}
mem = store(mem, c)

adr3 = "00001110111111"  # hex address: 7BF
cache = dir_map_cache(cache, adr3, mem)

def check_cache(cache, adr):
    block_label = adr[4:8]
    if block_label in cache and cache[block_label][2] == 1:
        print("Hit")
    else:
        print("Miss")
```

0s completed at 4:33 AM

Search 4:39 AM 12/8/2023

```
        print("Miss")

check_cache(cache, adr1)
check_cache(cache, adr2)
check_cache(cache, adr3)
```

```
Hit
Hit
Hit
```

#question no.3

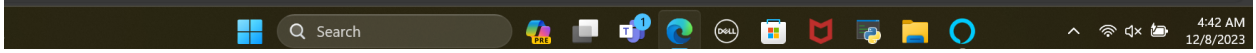
```python
#question no.3
import time

# Initialize memory
init_mem = {}

def store(storage, elm):
    storage.update(elm)
    return storage

# Initialize cache
cache = {
    "blk0": ["00000000000", [0,0,0,0,0,0,0,0], 0, 0],
    "blk1": ["00000000000", [0,0,0,0,0,0,0,0], 0, 0],
    "blk2": ["00000000000", [0,0,0,0,0,0,0,0], 0, 0],
    "blk3": ["00000000000", [0,0,0,0,0,0,0,0], 0, 0]
}

def fully_ass_cache(cache, adr, storage):
    # Extract tag from address
    tag = adr[:-3]

    # Check if the address is already in cache
    for block, data in cache.items():
        if data[0] == tag:  # Compare tags
            # Update LRU timestamp
```

```python
[47]    # Check if the address is already in cache
        for block, data in cache.items():
            if data[0] == tag:  # Compare tags
                # Update LRU timestamp
                data[3] = time.time()
                return cache

        # If not in cache, check if there's an empty line
        for block, data in cache.items():
            if data[2] == 0:  # Check valid bit
                # Store data in cache
                data[0] = tag  # Store tag
                data[1] = storage.get(tag, [0,0,0,0,0,0,0,0])  # Use tag as the key to retrieve block data
                data[2] = 1  # Set valid bit
                data[3] = time.time()  # Update LRU timestamp
                return cache

        # If cache is full, find the LRU block
        lru_block = min(cache.items(), key=lambda x: x[1][3])[0]
        # Replace the LRU block
        cache[lru_block][0] = tag  # Store new tag
        cache[lru_block][1] = storage.get(tag, [0,0,0,0,0,0,0,0])  # Use tag as the key to retrieve new block data
        cache[lru_block][2] = 1  # Set valid bit
        cache[lru_block][3] = time.time()  # Update LRU timestamp

        return cache
```

0s completed at 4:33 AM

```
# Storing data in memory
a = {"00000110101000": [0,1,2,3,4,5,6,7]}
init_mem = store(init_mem, a)

b = {"00001110101000": [10,11,12,13,14,15,16,17]}
init_mem = store(init_mem, b)

c = {"00011110101000": [20,21,22,23,24,25,26,27]}
init_mem = store(init_mem, c)

d = {"00111110101000": [30,31,32,33,34,35,36,37]}
init_mem = store(init_mem, d)

e = {"01111110101000": [40,41,42,43,44,45,46,47]}
init_mem = store(init_mem, e)

# Using the fully associative cache
adr1 = "00000110101010" # hex address: 1AA
cache = fully_ass_cache(cache, adr1, init_mem)

adr2 = "00001110101010" # hex address: 3AA
cache = fully_ass_cache(cache, adr2, init_mem)

adr3 = "00011110101111" # hex address: 7AF
cache = fully_ass_cache(cache, adr3, init_mem)

adr4 = "00111110101101" # hex address: FAD
```

0s completed at 4:33 AM




```
e = {"01111110101000": [40,41,42,43,44,45,46,47]}
init_mem
```

def fully_ass_cache(cache, adr, storage)

View source

<function fully_ass_cache at 0x7a91f1c35f30>

```
# Using
adr1 = "
cache = fully_ass_cache(cache, adr1, init_mem)

adr2 = "00001110101010" # hex address: 3AA
cache = fully_ass_cache(cache, adr2, init_mem)

adr3 = "00011110101111" # hex address: 7AF
cache = fully_ass_cache(cache, adr3, init_mem)

adr4 = "00111110101101" # hex address: FAD
cache = fully_ass_cache(cache, adr4, init_mem)

adr5 = "01111110101110" # hex address: 1FAE
cache = fully_ass_cache(cache, adr5, init_mem)

# Print the final state of the cache
print(cache)
```

```
{'blk0': ['01111110101', [0, 0, 0, 0, 0, 0, 0, 0], 1, 1702354661.960095], 'blk1': ['00001110101', [0, 0, 0, 0, 0, 0, 0, 0], 1, 1702354661.959951
```