Suraj Lamichhane
19708
Computer Organization
Assignment 5 ScreenShoots:

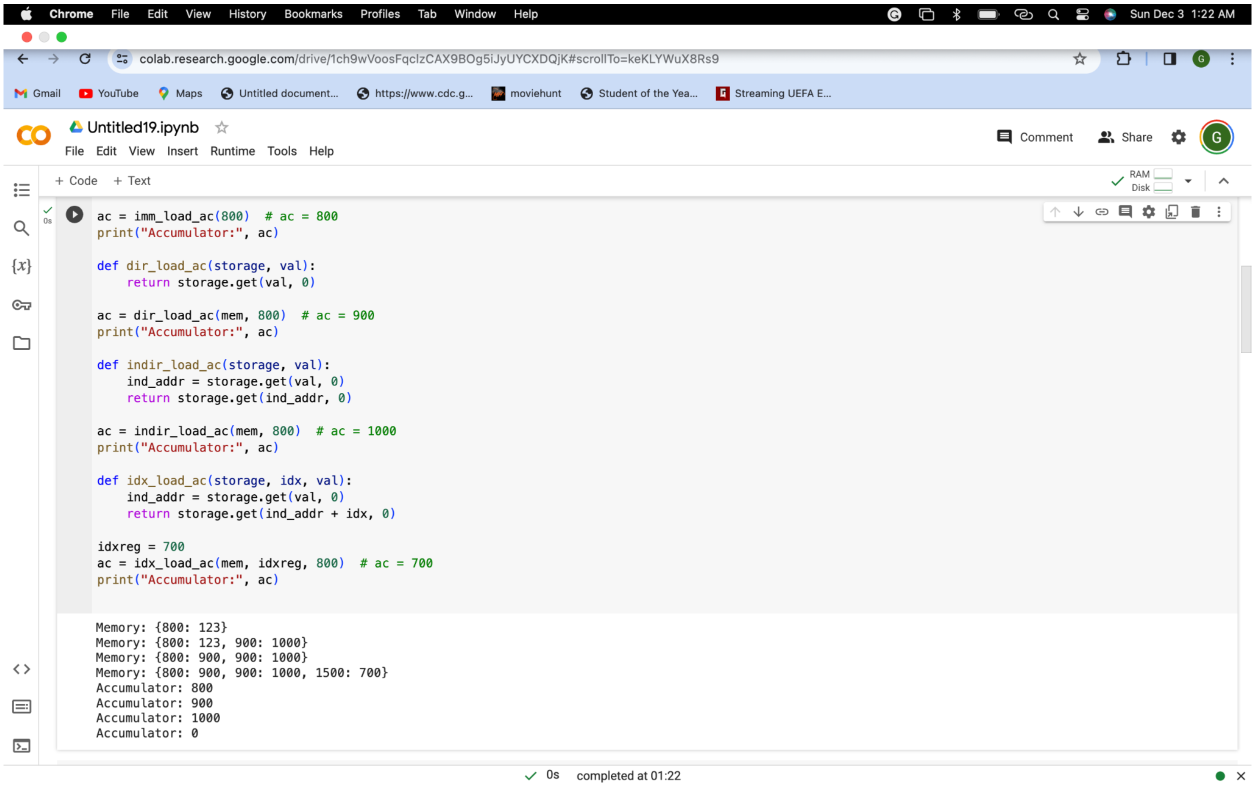
Qno.1)

Code:

```
# Qno.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):
   storage.update(elm)
   return storage

mem = store(init_mem, a)  # mem = {800: 123}
print("Memory:", mem)

mem = store(mem, b)  # mem = {800: 123, 900: 1000}
print("Memory:", mem)

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

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

def imm_load_ac(val):
   return val

ac = imm_load_ac(800)  # ac = 800
```

```
print("Accumulator:", ac)

def dir_load_ac(storage, val):
   return storage.get(val, 0)

ac = dir_load_ac(mem, 800)  # ac = 900
print("Accumulator:", ac)

def indir_load_ac(storage, val):
   ind_addr = storage.get(val, 0)
   return storage.get(ind_addr, 0)

ac = indir_load_ac(mem, 800)  # ac = 1000
print("Accumulator:", ac)

def idx_load_ac(storage, idx, val):
   ind_addr = storage.get(val, 0)
   return storage.get(ind_addr + idx, 0)

idxreg = 700
ac = idx_load_ac(mem, idxreg, 800)  # ac = 700
print("Accumulator:", ac)
```

Output:


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Sun Dec 3 1:22 AM
colab.research.google.com/drive/1ch9wVoosFqclzCAX9BOg5iJyUYCXDQjK#scrollTo=keKLYWuX8Rs9
Gmail YouTube Maps Untitled document... https://www.cdc.g... moviehunt Student of the Yea... Streaming UEFA E...
Untitled19.ipynb
File Edit View Insert Runtime Tools Help
Comment Share
+ Code + Text
RAM Disk
0s completed at 01:22

```
ac = imm_load_ac(800)  # ac = 800
print("Accumulator:", ac)

def dir_load_ac(storage, val):
    return storage.get(val, 0)

ac = dir_load_ac(mem, 800)  # ac = 900
print("Accumulator:", ac)

def indir_load_ac(storage, val):
    ind_addr = storage.get(val, 0)
    return storage.get(ind_addr, 0)

ac = indir_load_ac(mem, 800)  # ac = 1000
print("Accumulator:", ac)

def idx_load_ac(storage, idx, val):
    ind_addr = storage.get(val, 0)
    return storage.get(ind_addr + idx, 0)

idxreg = 700
ac = idx_load_ac(mem, idxreg, 800)  # ac = 700
print("Accumulator:", ac)
```

```
Memory: {800: 123}
Memory: {800: 123, 900: 1000}
Memory: {800: 900, 900: 1000}
Memory: {800: 900, 900: 1000, 1500: 700}
Accumulator: 800
Accumulator: 900
Accumulator: 1000
Accumulator: 0
```

Qno.2)

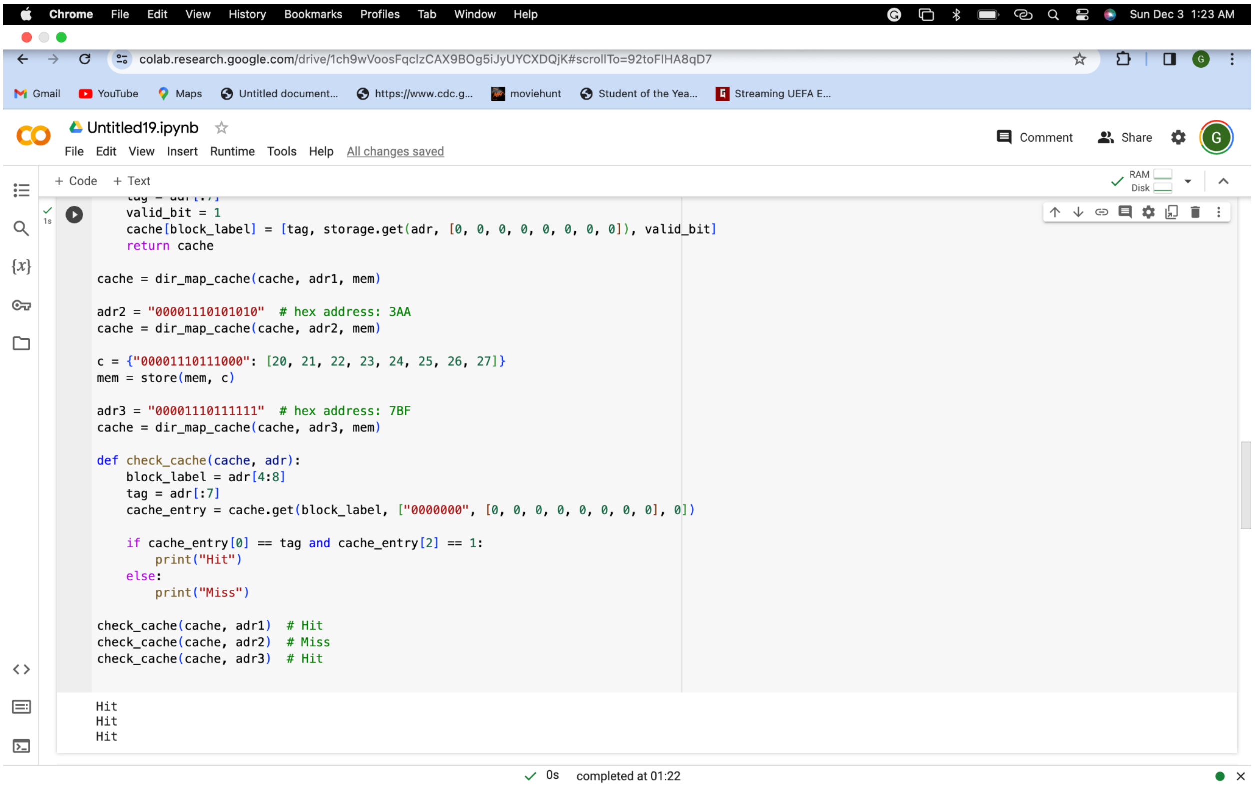
Code:

```
# Qno.2
init_mem = {}

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

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

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],
        "1110": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
        "1111": ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0]}

adr1 = "00000110101010"  # hex address: 1AA

def dir_map_cache(cache, adr, storage):
   block_label = adr[4:8]
   tag = adr[:7]
   valid_bit = 1
   cache[block_label] = [tag, storage.get(adr, [0, 0, 0, 0, 0, 0, 0, 0]),
valid_bit]
   return cache

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]
    tag = adr[:7]
    cache_entry = cache.get(block_label, ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0])

    if cache_entry[0] == tag and cache_entry[2] == 1:
        print("Hit")
    else:
        print("Miss")

check_cache(cache, adr1)  # Hit
check_cache(cache, adr2)  # Miss
check_cache(cache, adr3)  # Hit
```

Output:


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Sun Dec 3 1:23 AM
colab.research.google.com/drive/1ch9wVoosFqclzCAX9BOg5iJyUYCXDQjK#scrollTo=92toFIHA8qD7
Gmail
YouTube
Maps
Untitled document...
https://www.cdc.g...
moviehunt
Student of the Yea...
Streaming UEFA E...
Untitled19.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
0s completed at 01:22

```
    valid_bit = 1
    cache[block_label] = [tag, storage.get(adr, [0, 0, 0, 0, 0, 0, 0, 0]), valid_bit]
    return cache

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]
    tag = adr[:7]
    cache_entry = cache.get(block_label, ["0000000", [0, 0, 0, 0, 0, 0, 0, 0], 0])

    if cache_entry[0] == tag and cache_entry[2] == 1:
        print("Hit")
    else:
        print("Miss")

check_cache(cache, adr1)  # Hit
check_cache(cache, adr2)  # Miss
check_cache(cache, adr3)  # Hit
```

```
Hit
Hit
Hit
```

Qno.3)

Code:

```
#Qno.3
init_mem = {}

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

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

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

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

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

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

# Initialize cache
# Cache format: key -> block label
# Value -> tag(11 bits), values of 8 words, valid(1 bit)
# Assume that there are only 4 cache lines
cache = {
   "blk0": ["00000000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
   "blk1": ["00000000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
   "blk2": ["00000000000", [0, 0, 0, 0, 0, 0, 0, 0], 0],
   "blk3": ["00000000000", [0, 0, 0, 0, 0, 0, 0, 0], 0]
}

def fully_ass_cache(cache, adr, storage):
   block_label = adr[4:8]
   tag = adr[:11]
   valid_bit = 1

   # Find an available cache line or evict one if necessary
   for line in cache:
       if cache[line][2] == 0:
           cache[line] = [tag, storage.get(adr, [0, 0, 0, 0, 0, 0, 0, 0]),
valid_bit]
           return cache

   # If no available cache line, evict the first one (LRU policy)
   lru_line = min(cache, key=lambda x: cache[x][2])
   cache[lru_line] = [tag, storage.get(adr, [0, 0, 0, 0, 0, 0, 0, 0]),
valid_bit]
   return cache

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

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

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

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

adr5 = '01111110101110'  # hex address: 1FAE
cache = fully_ass_cache(cache, adr5, mem)

print(cache)
```

Output: