香 港 中 文 大 學
The Chinese University of Hong Kong

CSCI2510 Computer Organization
Tutorial 11: Review for Final

Tsun-Yu YANG

yangty@cse.cuhk.edu.hk

Outline

- Assignment 3 Solution
  - Q1
  - Q2
  - Programming Exercises

Assignment 3 Solution – Q1

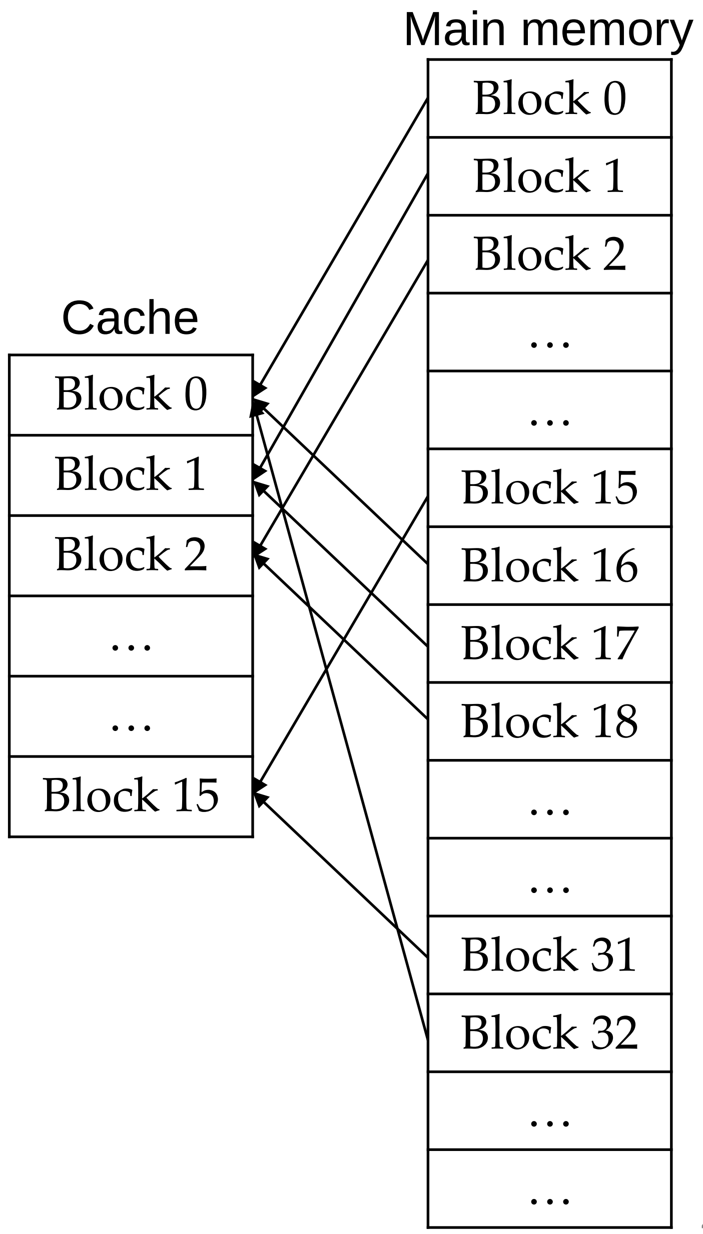
- Consider a computer uses 32-bit byte-addressable memory addresses. Assume it has a memory of 1G bytes and has a one-level cache of 1K bytes with 64 bytes per block and 4 bytes per word.

  - (a) Derive the number of bits required by different fields (i.e., Tag, Block, Set, Word, and Byte) for the following types of cache mapping functions:
    1) direct mapping,
    2) associative mapping, and
    3) two-way set-associative mapping.

Assignment 3 Solution – Q1(a)

1) For direct mapping

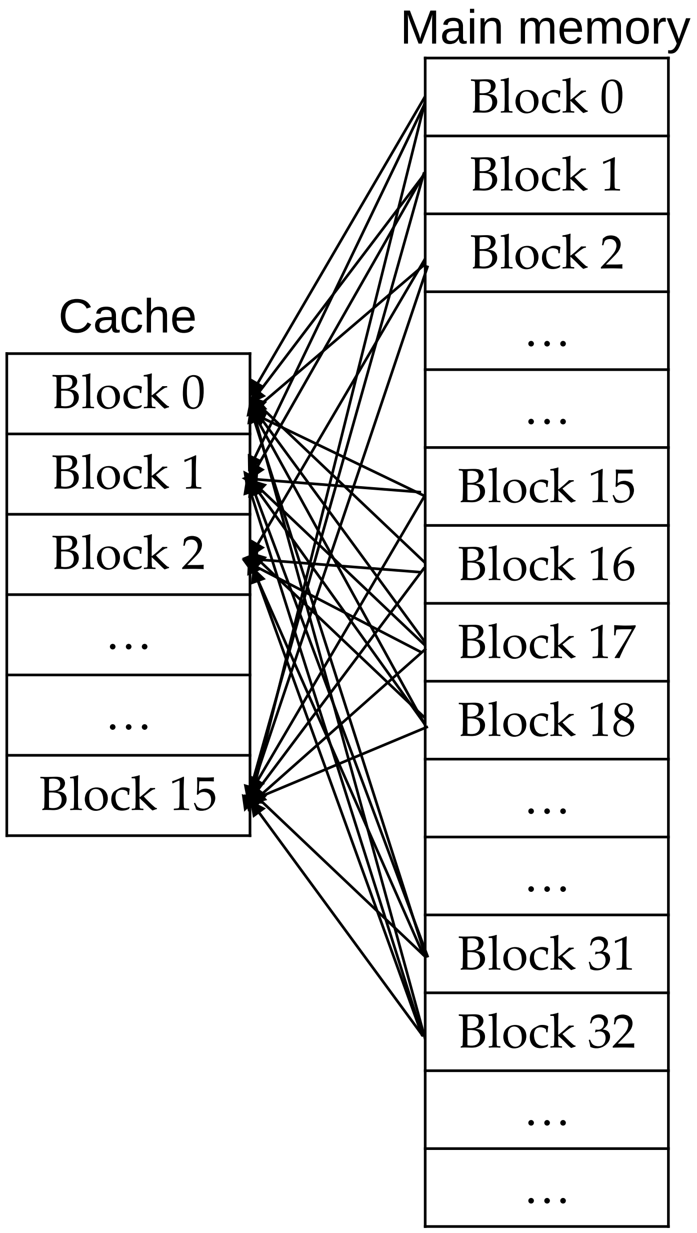
- Byte: Select one byte in a word
  - 2 bits: There're 4 = 2^2 B in a word.
- Word: Select one word in a block.
  - 4 bits: There're $\frac{64B/block}{4B/word} = 16 = 2^4\ word$ in a block.
- Block: Determine the CB in cache.
  - 4 bits: There're : $\frac{1KB}{64B/block} = 16\ block = 2^4\ block$ in cache.
- Tag: Keep track of which MB is placed in the corresponding CB.
  - 22 bits: **32** – (4 + 4 + 2) = 22 bits.



Assignment 3 Solution – Q1(a)

2) For associative mapping

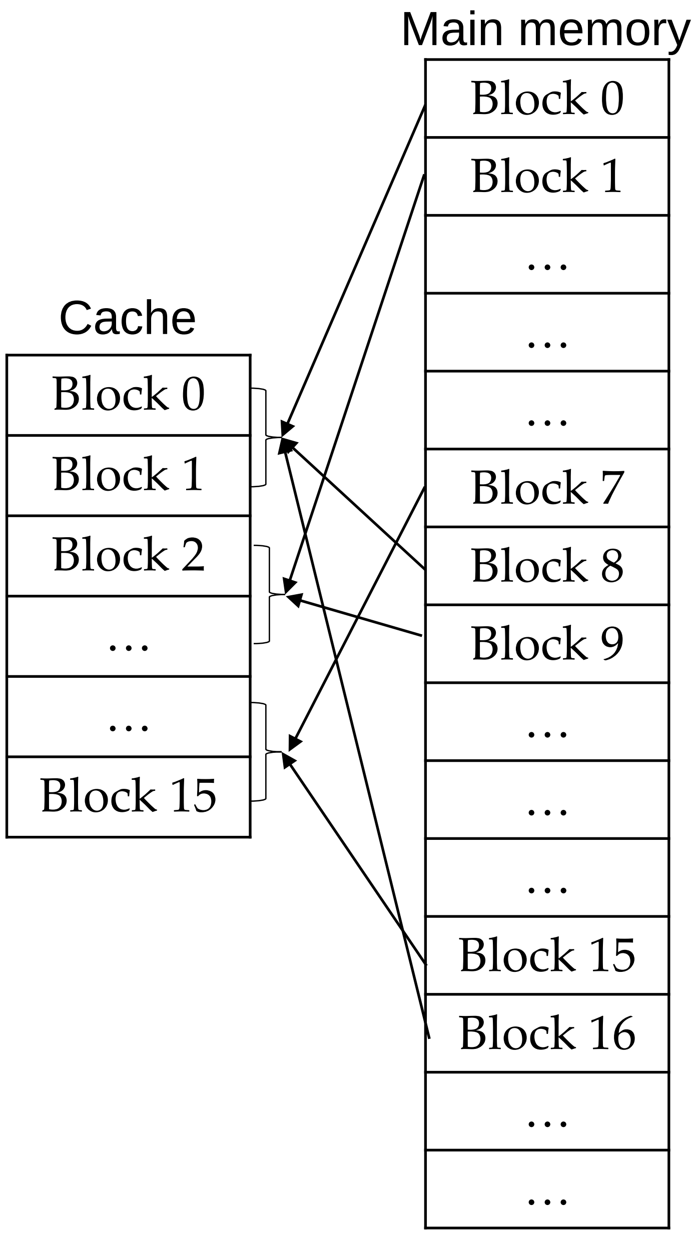
- Byte: Select one byte in a word.
  - 2 bits: There're $4 = 2^2$ B in a word.
- Word: Select one word in a block.
  - 4 bits: There're $\frac{64B/block}{4B/word} = 16 = 2^4\ word$ in a block.
- Tag: Keep track of which MB is placed in the corresponding CB.
  - 26 bits: **32** – (4 + 2) = 26 bits.



Assignment 3 Solution – Q1(a)

3) For 2-way set-associative mapping

- Byte: Select one byte in a word.
  - 2 bits: There're 4 = 2^2 B in a word.
- Word: Select one word in a block.
  - 4 bits: There're $\frac{64B/block}{4B/word} = 16 = 2^4\ word$ in a block.
- Set: Determine the CB in cache.
  - 3 bits: There're : $\frac{16block}{2block/set} = 8 = 2^3\ set$ in cache.
- Tag: Keep track of which MB is placed in the corresponding CB.
  - 23 bits: **32** – (3 + 4 + 2) = 23 bits.



Assignment 3 Solution – Q1(b)

(b) Suppose that the processor fetches 320 consecutive words (i.e. 20 blocks) starting at memory location 0, and **repeats** this fetch sequence for one more time. Assume that the cache is initially empty, derive the cache hit rate(s) for:

1) direct mapping,
2) associative mapping (with LRU), and
3) two-way set-associative mapping (with LRU).

Assignment 3 Solution – Q1(b)

- For seqentially read, there are two situations:

I. When the target block is not in cache
   - Access word 0, miss
   - Load the block containing word 0 ~ word 15 to cache
   - Access word 1 ~ word 15, all hit

II. When the target block is in cache
   - Access word 0 ~ word 15, all hit

As a result,

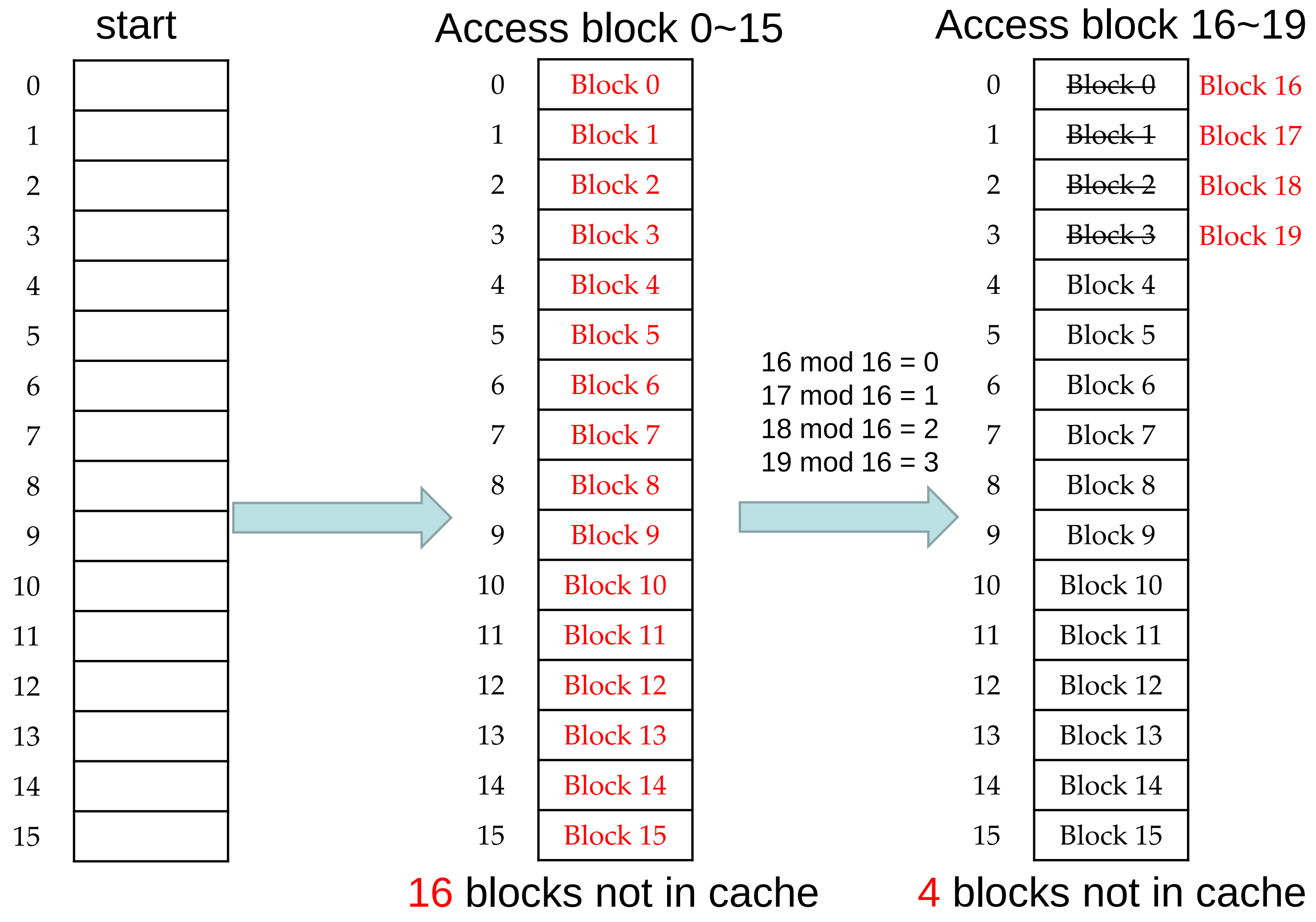
If the block is not in cache → 1 miss + 15 hit

If the block is in cache → 16 hit

Assignment 3 Solution – Q1(b)

1) Direct mapping

Round 1



	start	Access block 0~15	Access block 16~19
0		Block 0	~~Block 0~~ Block 16
1		Block 1	~~Block 1~~ Block 17
2		Block 2	~~Block 2~~ Block 18
3		Block 3	~~Block 3~~ Block 19
4		Block 4	Block 4
5		Block 5	Block 5
6		Block 6	Block 6
7		Block 7	Block 7
8		Block 8	Block 8
9		Block 9	Block 9
10		Block 10	Block 10
11		Block 11	Block 11
12		Block 12	Block 12
13		Block 13	Block 13
14		Block 14	Block 14
15		Block 15	Block 15

16 mod 16 = 0
17 mod 16 = 1
18 mod 16 = 2
19 mod 16 = 3

16 blocks not in cache

4 blocks not in cache

Assignment 3 Solution – Q1(b)

1) Direct mapping

Round 2

start

0	Block 16
1	Block 17
2	Block 18
3	Block 19
4	Block 4
5	Block 5
6	Block 6
7	Block 7
8	Block 8
9	Block 9
10	Block 10
11	Block 11
12	Block 12
13	Block 13
14	Block 14
15	Block 15

Access block 0~15

0	~~Block 16~~	Block 0
1	~~Block 17~~	Block 1
2	~~Block 18~~	Block 2
3	~~Block 19~~	Block 3
4	Block 4	
5	Block 5	
6	Block 6	
7	Block 7	
8	Block 8	
9	Block 9	
10	Block 10	
11	Block 11	
12	Block 12	
13	Block 13	
14	Block 14	
15	Block 15	

4 blocks not in cache
12 blocks in cache

16%16 = 0
17%16 = 1
18%16 = 2
19%16 = 3

Access block 16~19

0	~~Block 0~~	Block 16
1	~~Block 1~~	Block 17
2	~~Block 2~~	Block 18
3	~~Block 3~~	Block 19
4	Block 4	
5	Block 5	
6	Block 6	
7	Block 7	
8	Block 8	
9	Block 9	
10	Block 10	
11	Block 11	
12	Block 12	
13	Block 13	
14	Block 14	
15	Block 15	

4 blocks not in cache

Assignment 3 Solution – Q1(b)

1) Direct mapping

- Round 1: 20 blocks not in cache
- Round 2: 8 blocks not in cache, 12 blocks in cache.

As a result,

If the block is not in cache → 1 miss + 15 hit

If the block is in cache → 16 hit

# words which are hit

$$hit\ rate = \frac{\overbrace{12*16+(20+8)*15}}{\underbrace{320*2}} = 0.95625$$

Total # words accessed

Assignment 3 Solution – Q1(b)

2) Associative mapping (with LRU)

Round 1

	start	Access block 0~15		Access block 16~19	
0		Block 0	← LRU block	~~Block 0~~	Block 16
1		Block 1		~~Block 1~~	Block 17
2		Block 2		~~Block 2~~	Block 18
3		Block 3		~~Block 3~~	Block 19
4		Block 4		Block 4	
5		Block 5		Block 5	
6		Block 6		Block 6	
7		Block 7		Block 7	
8		Block 8		Block 8	
9		Block 9		Block 9	
10		Block 10		Block 10	
11		Block 11		Block 11	
12		Block 12		Block 12	
13		Block 13		Block 13	
14		Block 14		Block 14	
15		Block 15		Block 15	

The last used 4 block is CB0~CB3

16 blocks not in cache

4 blocks not in cache

Assignment 3 Solution – Q1(b)

2) Associative mapping (with LRU)

Round 2

	start	Access block 0~11		Access block 12~15		Access block 16~19	
0	Block 16	Block 16 ← LRU block		~~Block 16~~	Block 12	Block 12	
1	Block 17	Block 17		~~Block 17~~	Block 13	Block 13	
2	Block 18	Block 18		~~Block 18~~	Block 14	Block 14	
3	Block 19	Block 19		~~Block 19~~	Block 15	Block 15	
4	Block 4 ← LRU block	~~Block 4~~	Block 0	Block 0 ← LRU block		~~Block 0~~	Block 16
5	Block 5	~~Block 5~~	Block 1	Block 1		~~Block 1~~	Block 17
6	Block 6	~~Block 6~~	Block 2	Block 2		~~Block 2~~	Block 18
7	Block 7	~~Block 7~~	Block 3	Block 3		~~Block 3~~	Block 19
8	Block 8	~~Block 8~~	Block 4	Block 4		Block 4	
9	Block 9	~~Block 9~~	Block 5	Block 5		Block 5	
10	Block 10	~~Block 10~~	Block 6	Block 6		Block 6	
11	Block 11	~~Block 11~~	Block 7	Block 7		Block 7	
12	Block 12	~~Block 12~~	Block 8	Block 8		Block 8	
13	Block 13	~~Block 13~~	Block 9	Block 9		Block 9	
14	Block 14	~~Block 14~~	Block 10	Block 10		Block 10	
15	Block 15	~~Block 15~~	Block 11	Block 11		Block 11	

12 blocks not in cache

4 blocks not in cache

4 blocks not in cache

Assignment 3 Solution – Q1(b)

2) Associative mapping (with LRU)

- Round 1: 20 blocks not in cache
- Round 2: 20 blocks not in cache

As a result,

If the block is not in cache → 1 miss + 15 hit

If the block is in cache → 16 hit

$$hit\ rate = \frac{(20+20)*15}{320*2} = 0.9375$$

Assignment 3 Solution – Q1(b)

3) Two-way set-associative mapping (with LRU)

Round 1

start

set	block0	block1
0		
1		
2		
3		
4		
5		
6		
7		

Block ID mod 8 to get Set ID

Access block 0~15

set	block0	block1
0	Block 0	Block 8
1	Block 1	Block 9
2	Block 2	Block 10
3	Block 3	Block 11
4	Block 4	Block 12
5	Block 5	Block 13
6	Block 6	Block 14
7	Block 7	Block 15

16 blocks not in cache

Access block 16~19

set	block0	block1	
0	~~Block 0~~	~~Block 8~~	← Block 16
1	~~Block 1~~	~~Block 9~~	← Block 17
2	~~Block 2~~	~~Block 10~~	← Block 18
3	~~Block 3~~	~~Block 11~~	← Block 19
4	Block 4	Block 12	
5	Block 5	Block 13	
6	Block 6	Block 14	
7	Block 7	Block 15	

4 blocks not in cache

* The least recently used data in sets is underlined.

Assignment 3 Solution – Q1(b)

3) Two-way set-associative mapping (with LRU)

Round 2 (0~11)

start

set	block0	block1
0	Block 16	Block 8
1	Block 17	Block 9
2	Block 18	Block 10
3	Block 19	Block 11
4	Block 4	Block 12
5	Block 5	Block 13
6	Block 6	Block 14
7	Block 7	Block 15

Access block 0~7

set	block0	block1	
0	Block 16	~~Block 8~~	Block 0
1	Block 17	~~Block 9~~	Block 1
2	Block 18	~~Block 10~~	Block 2
3	Block 19	~~Block 11~~	Block 3
4	Block 4	Block 12	
5	Block 5	Block 13	
6	Block 6	Block 14	
7	Block 7	Block 15	

4 blocks not in cache
4 blocks in cache

Access block 8~11

set	block0	block1	
0	Block 16	Block 0	
1	Block 17	Block 1	
2	Block 18	Block 2	
3	Block 19	Block 3	
4	Block 4	~~Block 12~~	Block 8
5	Block 5	~~Block 13~~	Block 9
6	Block 6	~~Block 14~~	Block 10
7	Block 7	~~Block 15~~	Block 11

4 blocks not in cache

Assignment 3 Solution – Q1(b)

3) Two-way set-associative mapping (with LRU)

Round 2 (12~19)

Access block 8~11

set	block0	block1
0	Block 16	Block 0
1	Block 17	Block 1
2	Block 18	Block 2
3	Block 19	Block 3
4	Block 4	Block 8
5	Block 5	Block 9
6	Block 6	Block 10
7	Block 7	Block 11

Access block 12~19

set	block0	block1	
0	Block 16	Block 0	
1	Block 17	Block 1	
2	Block 18	Block 2	
3	Block 19	Block 3	
4	~~Block 4~~	~~Block 8~~	Block 12
5	~~Block 5~~	~~Block 9~~	Block 13
6	~~Block 6~~	~~Block 10~~	Block 14
7	~~Block 7~~	~~Block 11~~	Block 15

4 blocks not in cache
4 blocks in cache

Assignment 3 Solution – Q1(b)

3) Two-way set-associative mapping (with LRU)

- Round 1: 20 blocks not in cache
- Round 2: 12 blocks not in cache, 8 blocks in cache.

As a result,

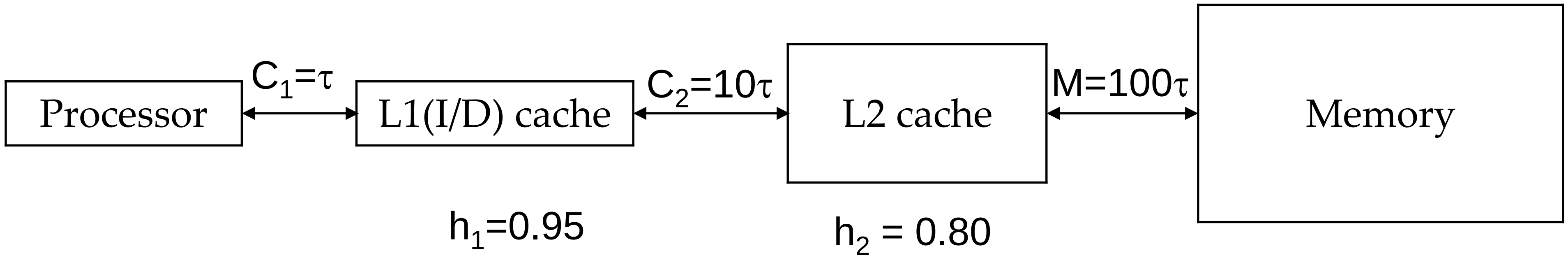
If the block is not in cache → 1 miss + 15 hit

If the block is in cache → 16 hit

$$hit\ rate = \frac{8 * 16 + (20 + 12) * 15}{320 * 2} = 0.95$$

Assignment 3 Solution – Q2

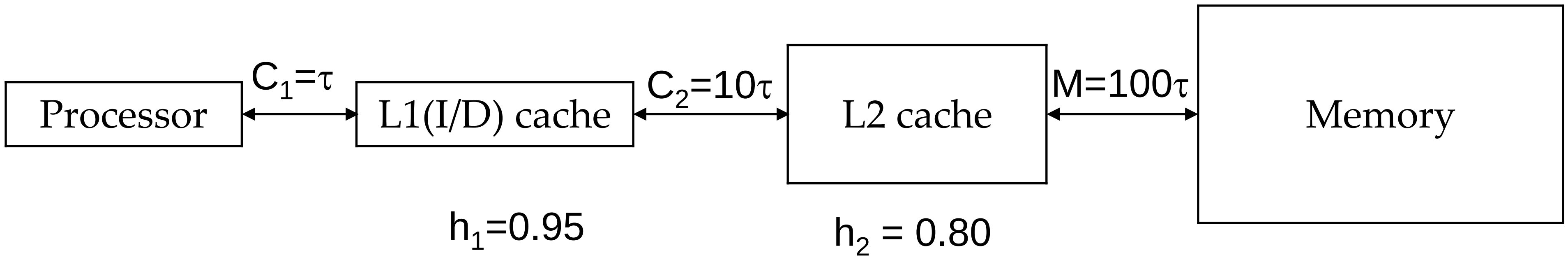
- Suppose that a computer has a processor with two L1 caches, one for instructions and one for data, and one L2 cache. Let τ be the access time for the two L1 caches, and it takes 10τ to access L2, and 100τ to access the main memory. Assume that the hit rates are the same for instructions and data and the hit rates in the L1 and L2 caches are 0.95 and 0.80, respectively.



Assignment 3 Solution – Q2(a)

(a) What fraction of accesses miss in both the L1 and L2 caches, thus requiring access to the main memory?

- $(1 - 0.95) * (1 - 0.80) = 0.01$

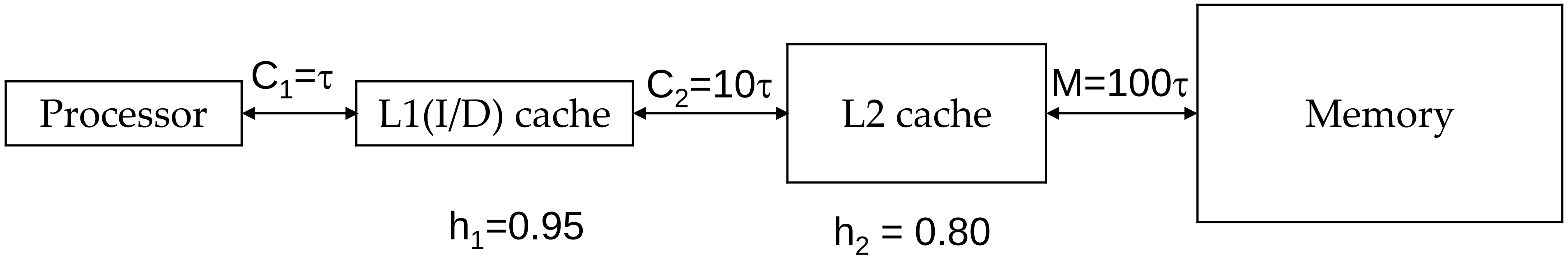


Assignment 3 Solution – Q2(b)

(b) What is the average memory access time as seen by the processor?

- $t_{avg} = \underline{0.95 * 1\ \tau} + \underline{0.05 * 0.8 * 12\ \tau} + \underline{0.05 * 0.2 * 122\tau}$

  $= 2.65\ \tau$



Assignment 1 Solution – Q2

(c) What hit rate of L1 Cache would be needed for reducing the average memory access time to 1.5 τ, if all other parameters are the same?

$$\underline{h_1 * 1\tau} + \underline{(1-h_1) * 0.8 * 12\tau} + \underline{(1-h_1) * 0.2 * 122\tau} = 1.5\tau$$

$$34 - 33 * h_1 = 1.5$$

$$h_1 = \frac{32.5}{33}$$

Can the same the average memory access time (i.e., 1.5 τ) be achieved by improving the hit rate of L2 cache?

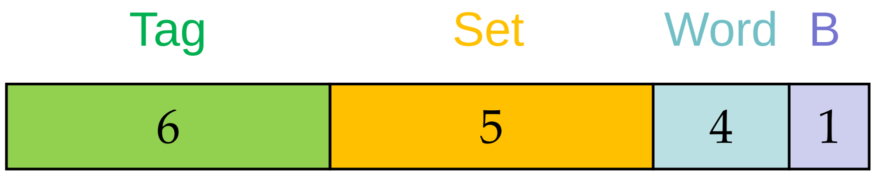
$$\underline{0.95 * 1\tau} + \underline{0.05 * h_2 * 12\tau} + \underline{0.05 * (1-h_2) * 122\tau} = 1.5\,\tau$$

$$5.5 * h_2 = 5.55$$

$$h_2 = \frac{5.55}{5.5} > 1$$ Impossible to be achieved

Programming Exercise 1

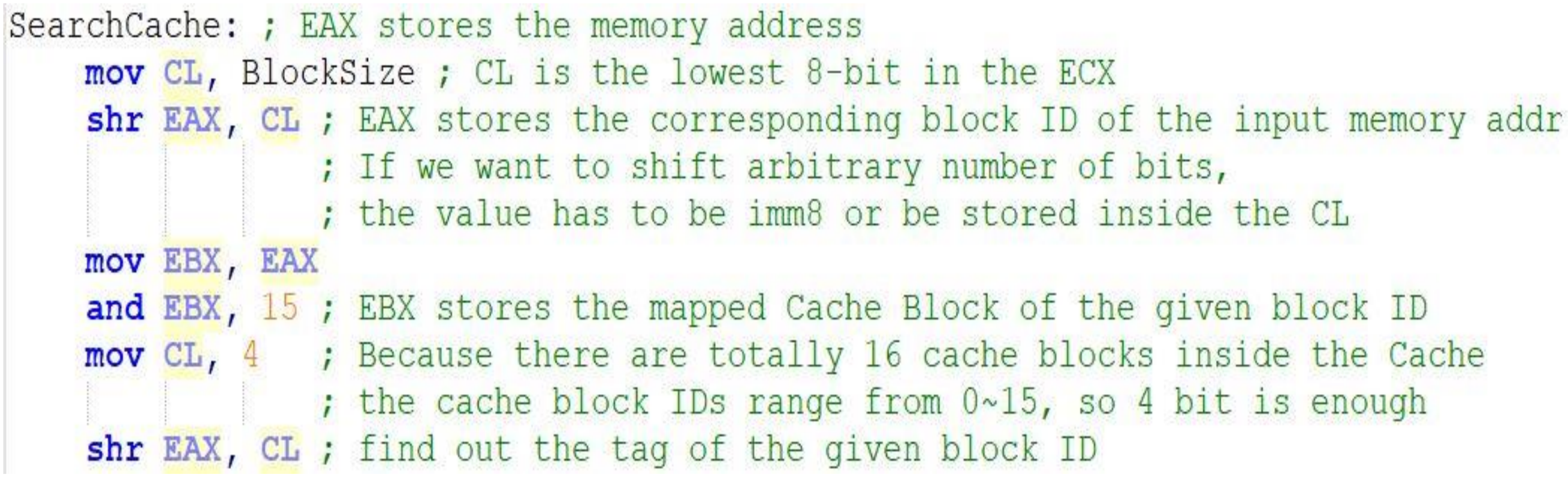
- In programming exercise 1, you are required to complete the **set_associative.asm** to implement a **four-way set-associative mapped cache**.
  - Each word is of 2 bytes (2^1) → 1 bit for B
  - Each block is of 16 words (2^4) → 4 bits for Word
  - # total cache bocks / 4-way = 32 (2^5) → 5 bits for Set
  - 16 – 1 – 4 – 5 = 6 bits for Tag



16-bit Main Memory Address

Exercise 1 – SearchSets

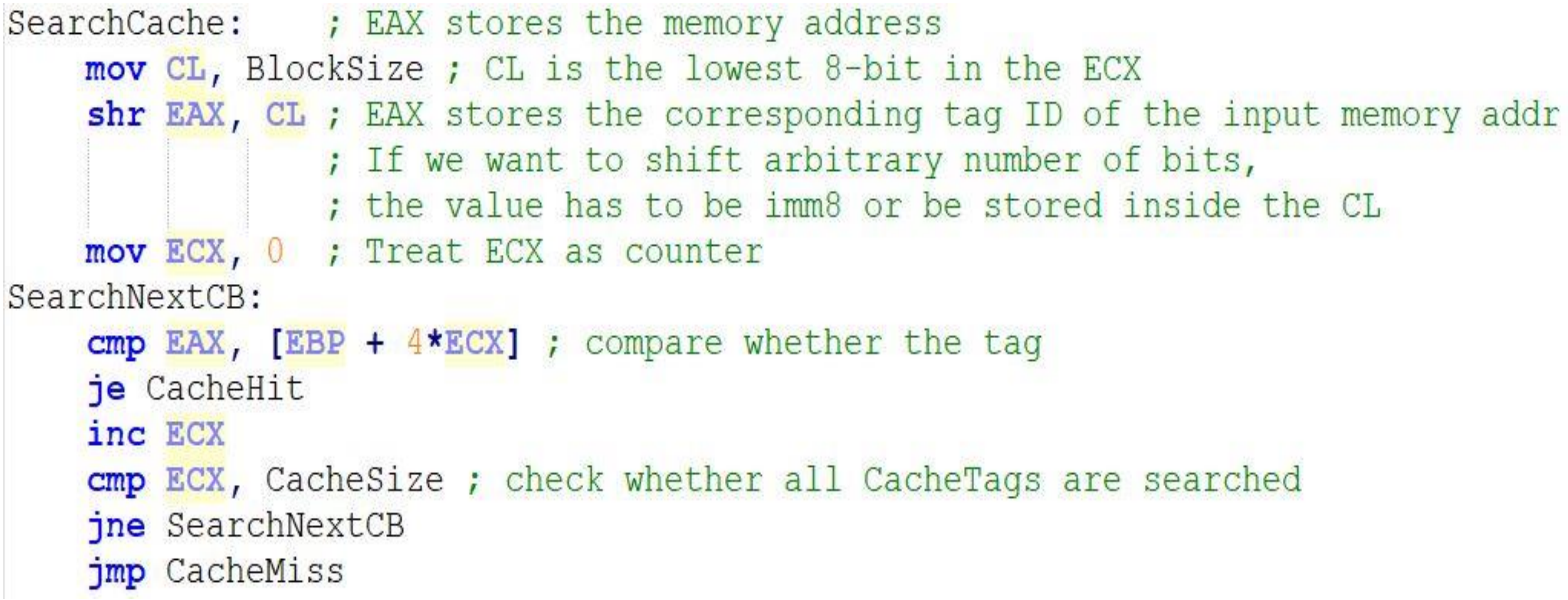
- Find out the set ID of the given memory address.
- You can find the similar code in the *SearchCache* of **Direct_Mapping.asm**.



```
SearchCache: ; EAX stores the memory address
    mov CL, BlockSize ; CL is the lowest 8-bit in the ECX
    shr EAX, CL ; EAX stores the corresponding block ID of the input memory addr
                ; If we want to shift arbitrary number of bits,
                ; the value has to be imm8 or be stored inside the CL
    mov EBX, EAX
    and EBX, 15 ; EBX stores the mapped Cache Block of the given block ID
    mov CL, 4   ; Because there are totally 16 cache blocks inside the Cache
                ; the cache block IDs range from 0~15, so 4 bit is enough
    shr EAX, CL ; find out the tag of the given block ID
```

Exercise 1 – SearchTags

- Search all tags within the set to find out it’s a cache hit or cache miss.
- You can find the similar code in *SearchCache* of **Associative_Mapping.asm**.



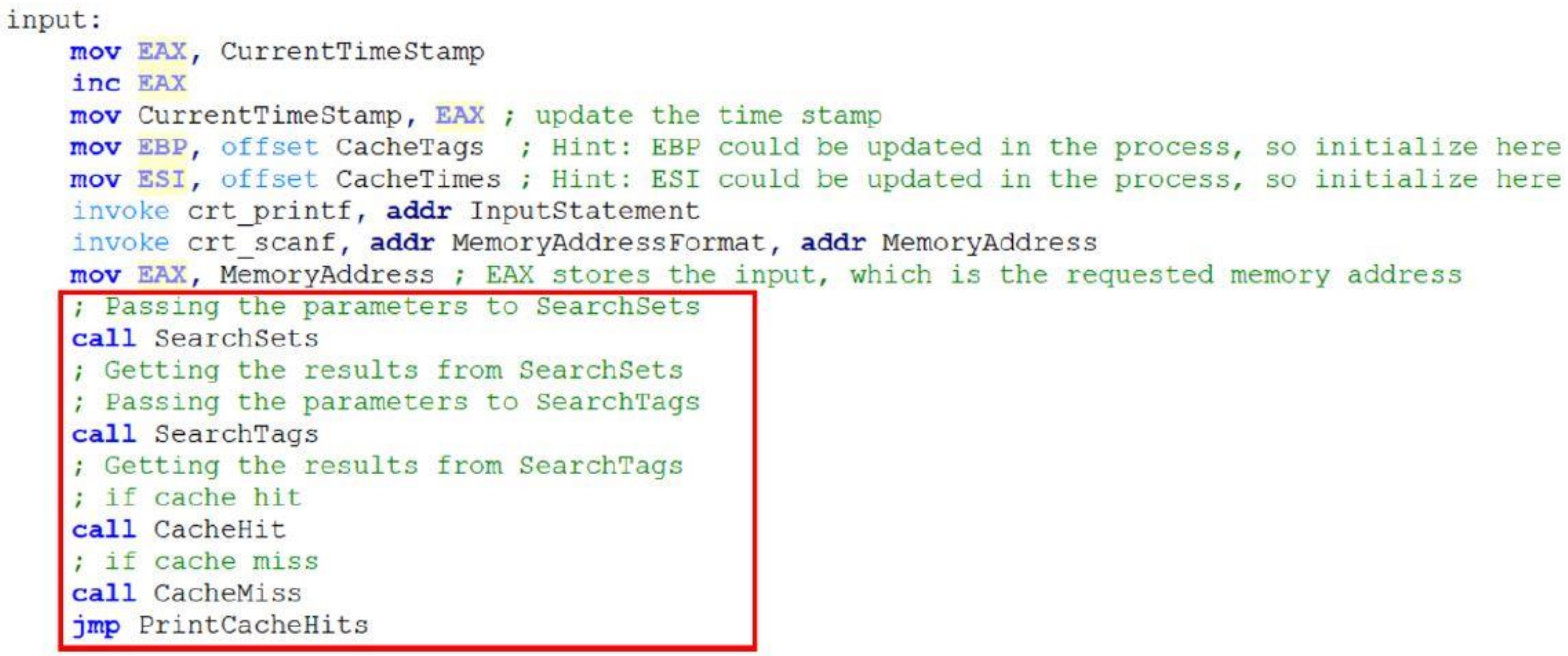
```
SearchCache:     ; EAX stores the memory address
    mov CL, BlockSize ; CL is the lowest 8-bit in the ECX
    shr EAX, CL ; EAX stores the corresponding tag ID of the input memory addr
                ; If we want to shift arbitrary number of bits,
                ; the value has to be imm8 or be stored inside the CL
    mov ECX, 0  ; Treat ECX as counter
SearchNextCB:
    cmp EAX, [EBP + 4*ECX] ; compare whether the tag
    je CacheHit
    inc ECX
    cmp ECX, CacheSize ; check whether all CacheTags are searched
    jne SearchNextCB
    jmp CacheMiss
```

Exercise 1 – CacheHit & CacheMiss

- The implementation of *CacheHit* & *CacheMiss* is related to the replacement algorithm.
- FIFO and LRU are very similar. The difference is the meanings of *CacheTimes* array.
  - LRU: *CacheTimes* stores the latest accessed time.
  - FIFO: *CacheTimes* stores the loaded time.
- For LRU:
  - Update the *CacheTimes* if *CacheHit*
  - Find the block with the smallest time stamp if *CacheMiss*
- For FIFO:
  - Do nothing if *CacheHit*
  - Find the block with the smallest time stamp if *CacheMiss*

Programming Exercise 2

- Convert the four main functions in **set_associative.asm** into subroutine
- Easy if being familiar with Tut07.



```
input:
    mov EAX, CurrentTimeStamp
    inc EAX
    mov CurrentTimeStamp, EAX ; update the time stamp
    mov EBP, offset CacheTags  ; Hint: EBP could be updated in the process, so initialize here
    mov ESI, offset CacheTimes ; Hint: ESI could be updated in the process, so initialize here
    invoke crt_printf, addr InputStatement
    invoke crt_scanf, addr MemoryAddressFormat, addr MemoryAddress
    mov EAX, MemoryAddress ; EAX stores the input, which is the requested memory address
    ; Passing the parameters to SearchSets
    call SearchSets
    ; Getting the results from SearchSets
    ; Passing the parameters to SearchTags
    call SearchTags
    ; Getting the results from SearchTags
    ; if cache hit
    call CacheHit
    ; if cache miss
    call CacheMiss
    jmp PrintCacheHits
```

Summary

- Assignment 3 Solution
  - Q1
  - Q2
  - Programming Exercises