
Microprocessor and Assembly Language

ASSIGNMENT 2

NAME: **Deepak Kumar**

REGISTRATION NUMBER: **SP21-BCS-017**

DATE: **25-10-2023**

SUBMITTED TO: **Mr. Taimur Shahzad**

SECTION: **BCS-6A**

Q1. What is Set-Associative Mapping? Discuss all the types of Set-Associative Mapping along with appropriate examples.

<u>SET-ASSOCIATIVE MAPPING</u>

- In computer systems, it is a memory mapping technique.
- RAM stores large amounts of data as compared to cache memory, but data retrieval is faster in cache memory. So, to effectively use RAM and cache memory together, set associative mapping is used.
- It maps a section of main memory (RAM) to a particular section of cache memory.
- The cache memory is divided into a collection of smaller cache blocks or sets.
- Each of the cache block or set is linked to a set of memory blocks.
- Each cache set has numerous cache blocks, and each memory block can map to any of the cache blocks in the appropriate set.

<u>TYPES</u>

Following are the two main types of Set-Associative Mapping:

1. Direct-Mapped Set-Associative Mapping

Each memory block can only map to one specific cache block within its corresponding set. The number of sets in the cache memory is equal to the number of memory blocks in the main memory divided by the associativity. Here, associativity is an integer power of two that represents number of cache blocks in each cache set. For example, if the main memory has 32 blocks and the associativity is 2, then there are 16 sets in the cache memory.

- **<u>Example</u>**

In the following example, there are 16 memory blocks in RAM with an associativity of 2. So, the cache memory is divided into 8 sets, each containing 2 cache blocks.

RAM	Cache Memory	
Block 0	Set 0	Cache Block 0
		Cache Block 1
Block 1	Set 1	Cache Block 2
		Cache Block 3
Block 2	Set 2	Cache Block 4
		Cache Block 5
Block 3	Set 3	Cache Block 6
		Cache Block 7
…		

Block 15	Set 7	Cache Block 30
		Cache Block 31

Here, Block 0 is mapped to Set 0, which contains 2 cache blocks: *Cache Block 0* and *Cache Block 1*. If Block 1 needs to be cached, it would be mapped to Set 1, which contains *Cache Block 2* and *Cache Block 3*. Similarly, remaining memory blocks are mapped to corresponding cache memory sets.

- **Advantage:**

Less hardware and management overhead to keep track tags associated with each cache block.

- **Disadvantage:**

Less flexibility in selecting a cache block to map against a memory block as each memory block can only map to one specific cache block.

2. Fully Associative Set-Associative Mapping

Each memory block can map to any cache block within the cache memory, and there are no sets. Each cache block is identified by a tag that corresponds to the memory block it is caching.

- **Example**

In the following example, there are N memory blocks in RAM that is mapped to N cache blocks identified by a tag. So, the total tags will be N as well.

RAM	Cache Memory	
Block 0	Cache Block 0	Tag 0
Block 1	Cache Block 1	Tag 1
Block 2	Cache Block 2	Tag 2
Block 3	Cache Block 3	Tag 3
…		
Block N	Cache Block N	Tag N

Here, *Block 0* can be cached in any of the cache blocks in the cache memory, which are identified by their respective tags.

- **Advantage:**

More flexibility in selecting a cache block to map against a memory block.

- **Disadvantage:**

More hardware and management overhead to keep track of tags associated with each cache block.

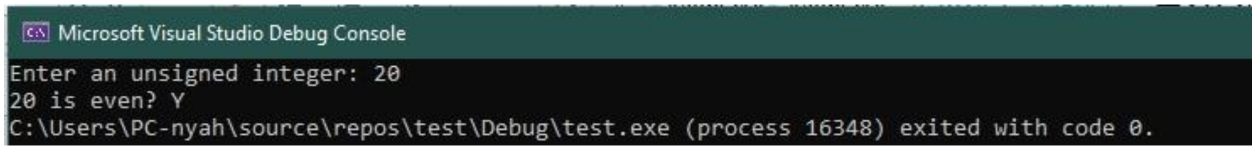
Q2. Write a program that finds out whether an unsigned integer in EAX register is even or odd. If the number is even, place 'Y' in DL otherwise 'N' in DL. (Hint: Least significant bit of an even number is always 0 while that of odd number is 1).

CODE

```
INCLUDE Irvine32.inc
 .data
inputPrompt BYTE "Enter an unsigned integer: ", 0  outputPrompt BYTE " is even? ", 0
 .code
main PROC
    ; Move address of input message to EDX register and display message on screen
    MOV EDX, OFFSET inputPrompt
    CALL WriteString

    ; Read unsigned decimal integer from the console and store it in EAX register
    CALL ReadDec
    ; Write value of EAX register on console
    CALL writeDec

    ; Move address of output message to EDX register and display message on screen
    MOV EDX, OFFSET outputPrompt
    CALL WriteString

    ; Get LSB of unsigned integer using Bitwise AND operator and 1
    AND EAX, 1

    ; Compare LSB with 0
    CMP EAX, 0

    ; If LSB is 0 i.e. comparison is true, jump to is_even label
    JE is_even
    ; Else move 'N' to the DL register
    MOV DL, 'N'

     ; Jump to the done label
    JMP done
```

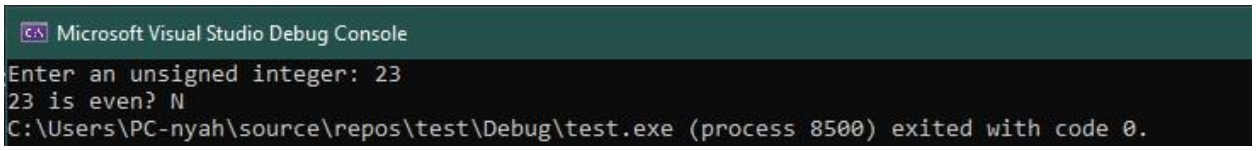
```
is_even:
    MOV DL, 'Y'         ; Move the value 'Y' to the DL register
 done:

    MOV AL, DL          ; Move the value of the DL register to the AL register
    CALL WriteChar      ; Display the result on the console
EXIT
  main ENDP
END main
```

OUTPUT



Microsoft Visual Studio Debug Console

```
Enter an unsigned integer: 20
20 is even? Y
C:\Users\PC-nyah\source\repos\test\Debug\test.exe (process 16348) exited with code 0.
```



Microsoft Visual Studio Debug Console

```
Enter an unsigned integer: 23
23 is even? N
C:\Users\PC-nyah\source\repos\test\Debug\test.exe (process 8500) exited with code 0.
```