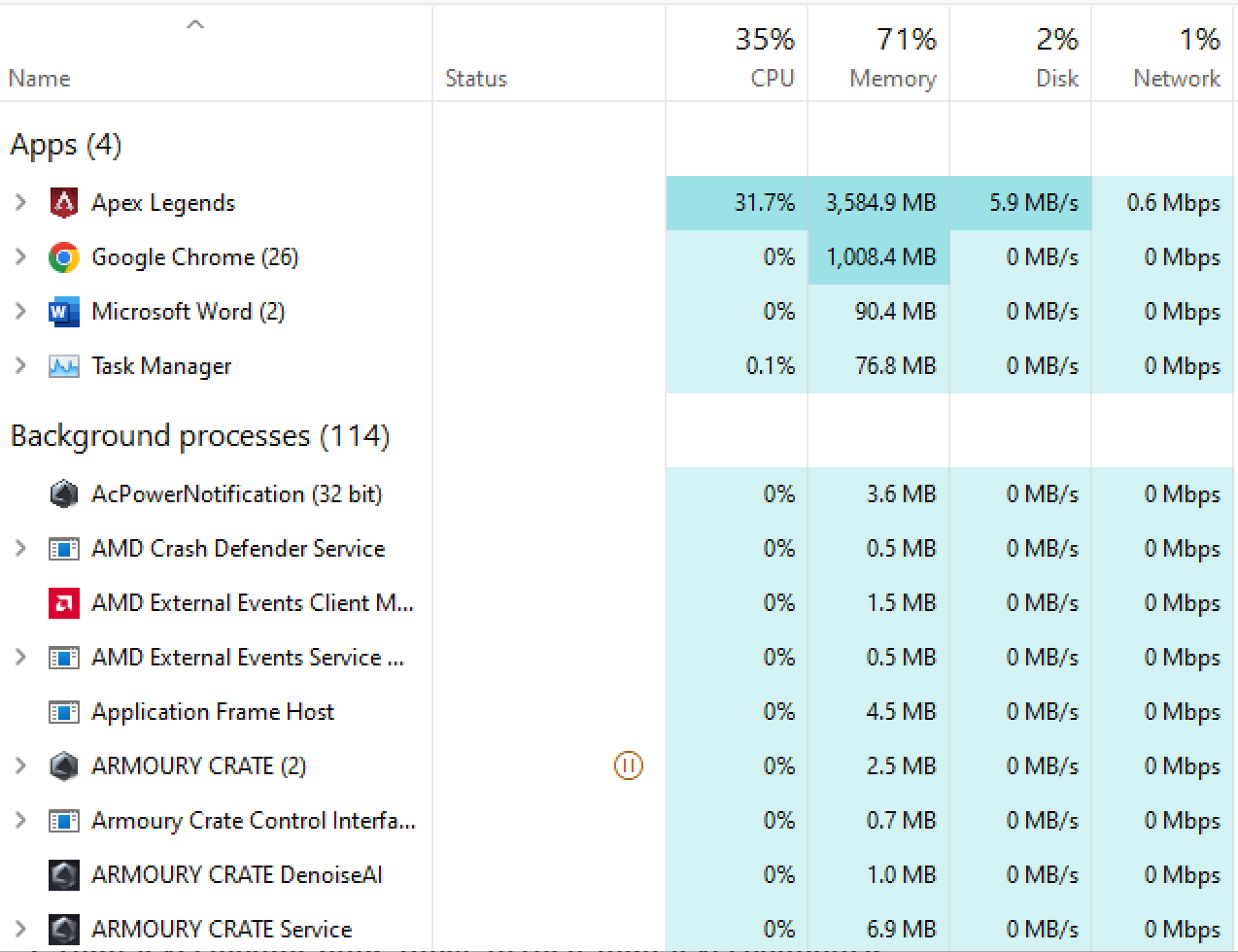
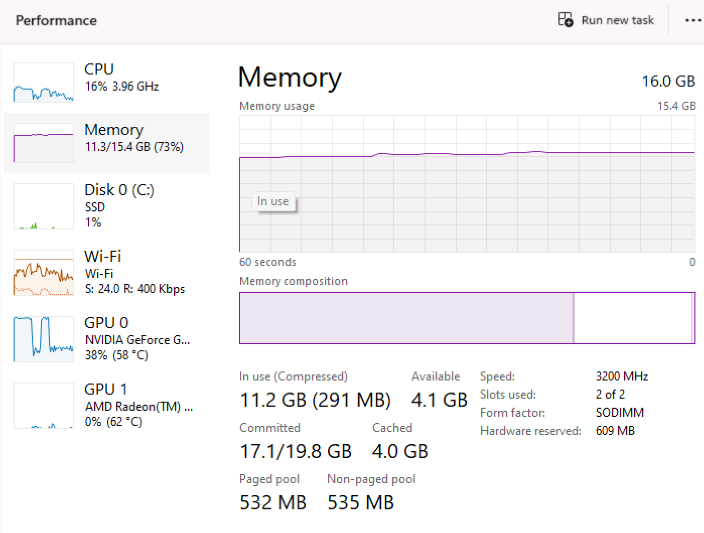
|  |  |
| --- | --- |
| **Name: Teh Jia Xuan** | **Student ID: 32844700** |
| **FIT1047 Assignment 2 – Processes and MARIE Programming** | |

**Part 1: Processes**

**1a. Memory usage of a process**

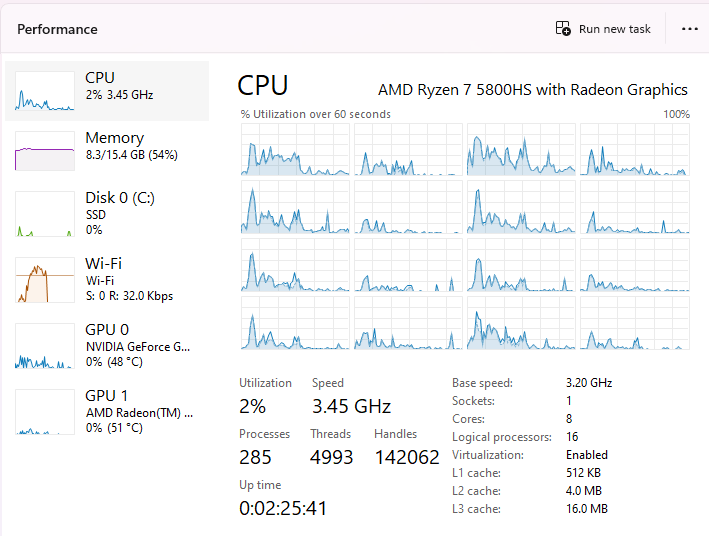
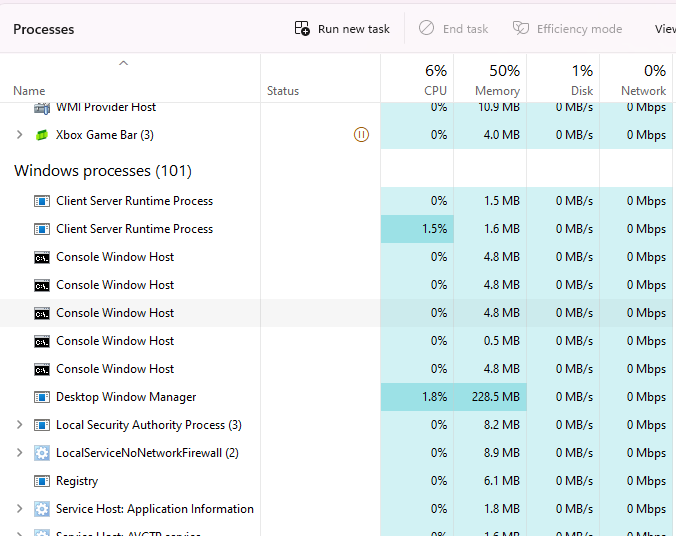
Random Access Memory(RAM) stores temporary memory where the task manager helps to show how much physical and virtual memory is available and how much each program uses it. In figure 1.1, the memory column refers to the percentage of the memory I have used. Under the Memory column, specifies how much memory has been “eaten” by each application. Likewise, Apex Legends uses approximately 3.5GB of memory. The fact is that when games are started, they will read and write data to RAM instead of keeping retrieved from a hard disk and causing inefficiency. Furthermore, google chrome also consume a huge amount of RAM due to chrome splitting every tab into a process. Hence, if a tab crashes, it won’t affect the entire application.



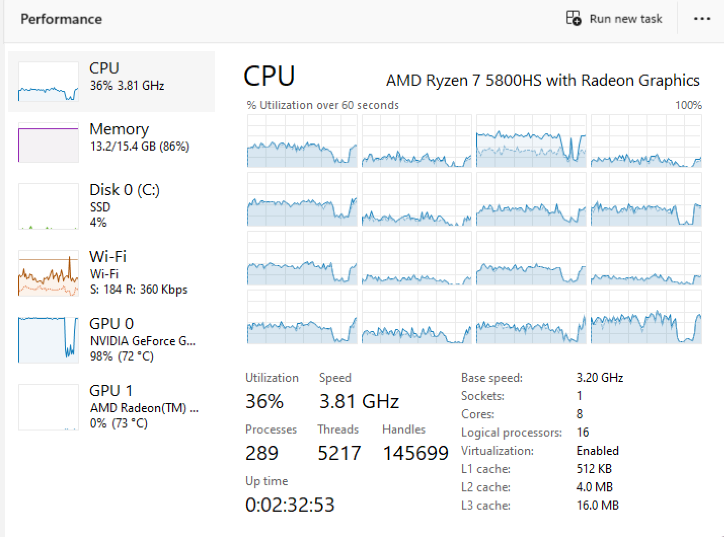
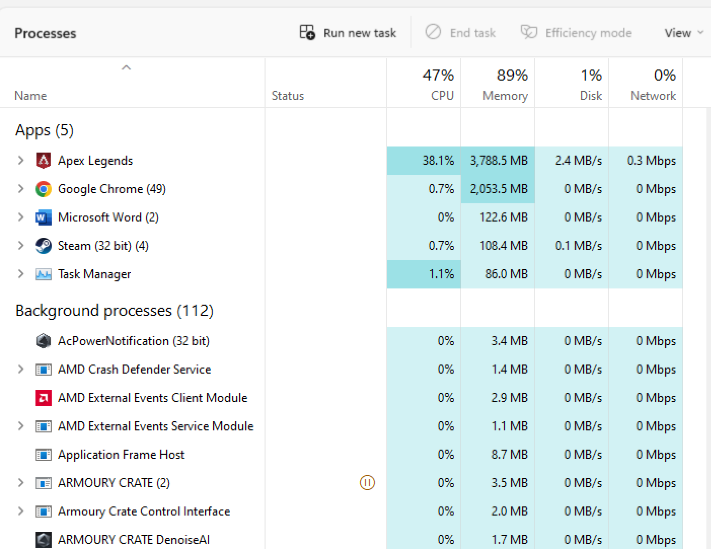
**Figure 1.1**  **Figure 1.2**

**1b. CPU usage of a process**

On the other hand, the task manager shows the percentage of CPU processes used, the clock speed and each cores usage by dividing the graph into 4x4 graphs. Likewise, each graph represents the corresponding cores. Moreover, the clock speed will increase depending on the workload of the CPU. Thus, the clock speed and the graph will increase if an application uses a lot of CPU. For instance, before “apex legends” run, CPU usage is 6%, the clock speed is 3.45GHz, and the graphs are stable. After “apex legends” run, CPU usage increased to 47% , clock speed increased to 3.81 GHz, and the graph climbed up instantly as there were many calculations needed to be done for the application. According to figure 1.5, 38.1% of CPU has been used by Apex Legends. On the contrary, some processes have less CPU usage, like Desktop Window Manager uses 1.8% of the CPU, and Client Server Runtime Process uses 1.5% of the CPU.

****

**Figure 1.3 Figure 1.4**

****

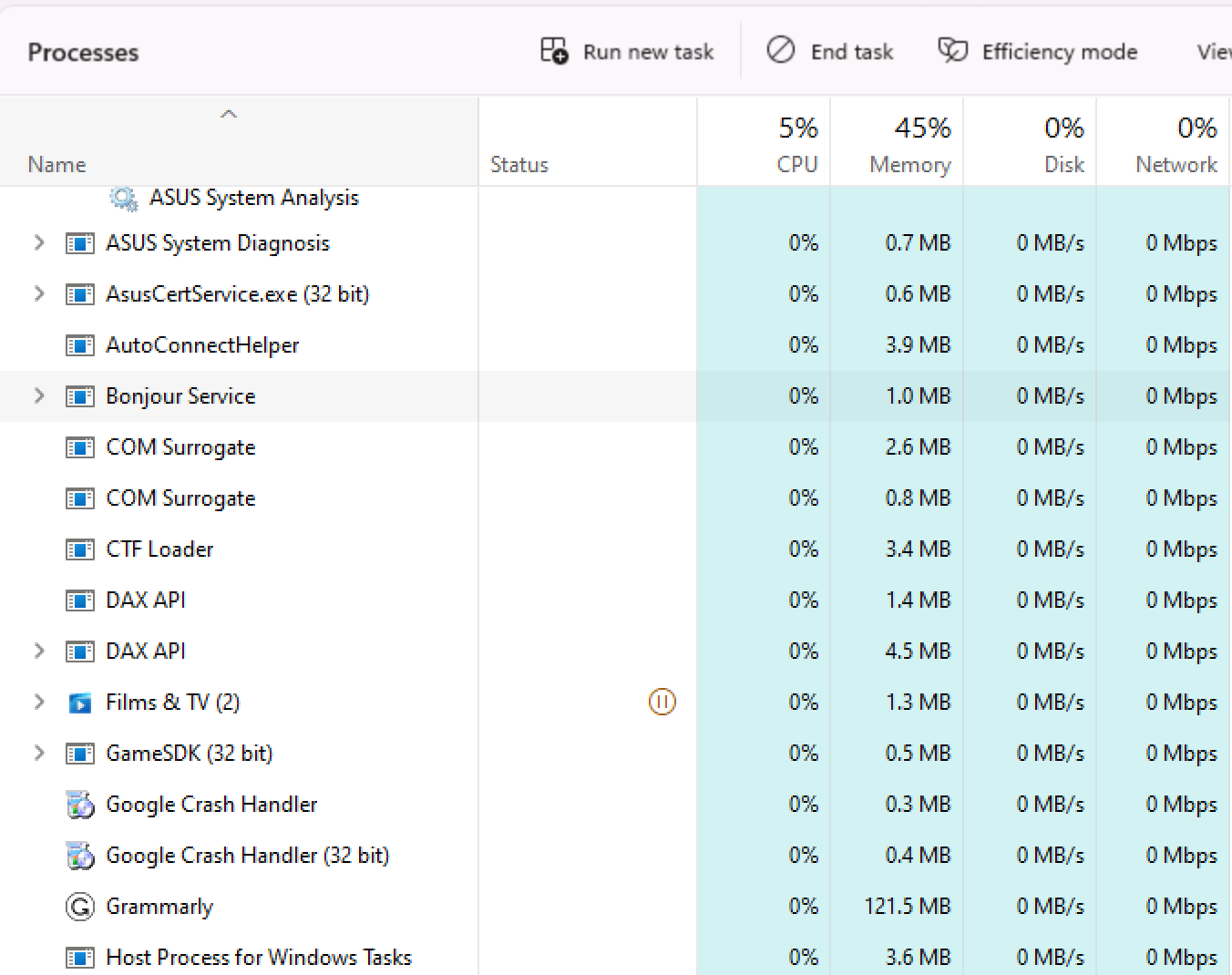
**Figure 1.5 Figure 1.6**

**Memory usage compared to RAM installed.**

The memory available will be less than the RAM installed in the computer. Due to the fact that, the operating system keeps a portion of ram for its usage. Thus, although I have 16GB RAM installed on my computer, only 15.4GB is available. The overall memory usage of all processes in my computer is 11.2GB and the graph of memory in task manager remain stable. Hence, 16GB RAM is sufficient for my usage.

**2.** Bonjour service is one of the processes that run on my computer. It exists because apple devices can communicate with each other easily, but when it comes to Mac and Windows. It is hard to enable these two systems to share data and services. Thus, bonjour services are created to enable Apple products to communicate with non-Apple products.

Bonjour service is a small application that silently works in the background, so it doesn’t use much internet and disk space. It enables users to set up a local area network(LAN) without configuration. For instance, Bonjour enables Apple and Windows OS to share resources and communicate without configuration settings. With the help of Bonjour service, it enables us to connect to other devices like scanners, printers etc. Bonjour service uses link-local addressing to assign IP addresses instead of Dynamic Host Configuration Protocol (DHCP). It works perfectly fine with IPv4 and IPv6 addressing. Furthermore, apps like iTunes and Safari use Bonjour service to communicate with devices.

****

**Figure 1.7**

**3.** The operating system (OS) must manage the files stored on a computer to protect the users from other processes (eg. buggy code, malicious) accessing our files, which could expose privacy to the public. Thus, the OS act as a middleman between the application and the hardware. Besides that, OS hides complexity for the user and provides a user-friendly interface as they don’t need to know what is running in the background and which processes run simultaneously. Thus, this provides a user-friendly interface to the user. Moreover, OS manages the files to enable programmers to access system resources easily.

**Part 2: MARIE Programme**

**Task 2.1: Printing name**

Test Case: TehJiaXuan

****

**Figure 2.1.1 Printing Name**

**Task 2.2: Subroutine prints out string**

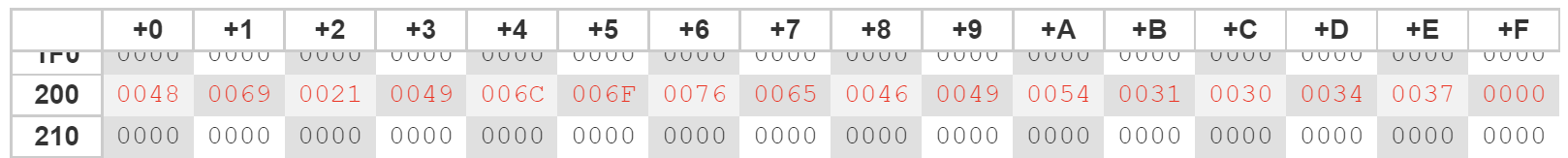
Test Case: TehJiaXuan

****

**Figure 2.2.1 Printing String**

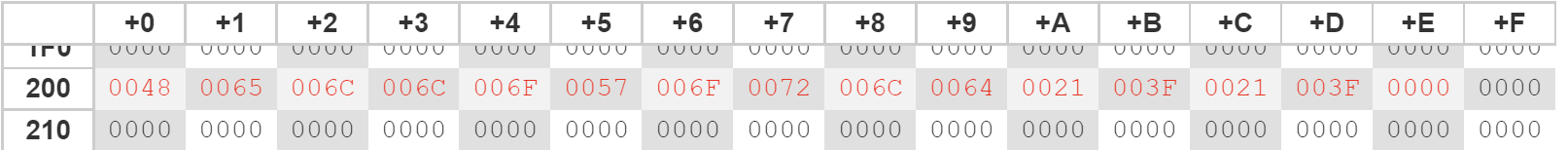
**Task 2.3: Subroutine input string from user**

Test Case 1 : Hi!IloveFIT1047

****

**Figure 2.3.1 Input String memory**

Test Case 2 : HelloWorld!?!?

****

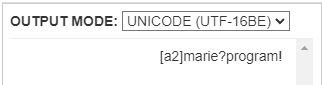
**Figure 2.3.2 Input String memory**

**Task 2.4: Subroutine convert to lowercase**

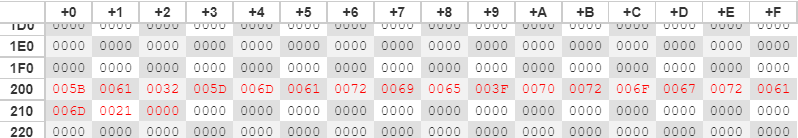
Test Case 1:

Input String: [A2]MARIE?Program!

Output String: [a2]marie?program!

****

**Figure 2.4.1 Convert lowercase**



**Figure 2.4.2 Convert lowercase memory**

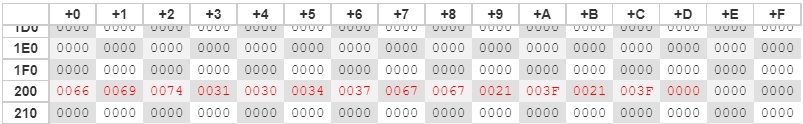
Test Case 2:

Input String: FIT1047gg!?!?

Output String: fit1047gg!?!?



**Figure 2.4.3 Convert lowercase**



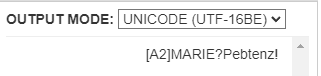
**Figure 2.4.4 Convert lowercase memory**

**Task 2.5: Subroutine ROT13 encode**

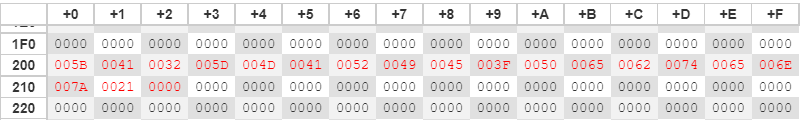
Test Case 1:

Input String: [A2]MARIE?Program!

Output String: [A2]MARIE?Pebtenz!

****

**Figure 2.5.1 ROT13 String**

****

**Figure 2.5.2 Memory ROT13 String**

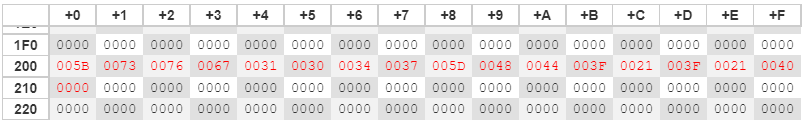
Test Case 2:

Input String: [FIT1047]HD?!?!@

Output String: [svg1047]HD?!?!@

****

**Figure 2.5.3 ROT13 String**

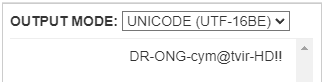
****

**Figure 2.5.4 Memory ROT13 String**

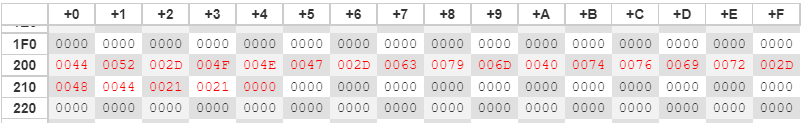
Test Case 3:

Input String: DR-ONG-plz@give-HD!!

Output String: DR-ONG-cym@tvir-HD!!

****

**Figure 2.5.5 ROT13 String**

****

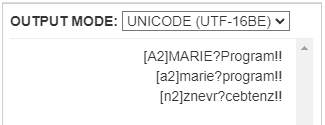
**Figure 2.5.6 Memory ROT13 String**

**Task 2.6: Nested subroutines**

Test Case 1:

Input String: [A2]MARIE?Program!!

Output String: [A2]MARIE?Program!!  
 [a2]marie?program!!  
 [n2]znevr?cebtenz!!

****

**Figure 2.6.1 Convert to encode string.**

****

**Figure 2.6.2 Memory encode string**

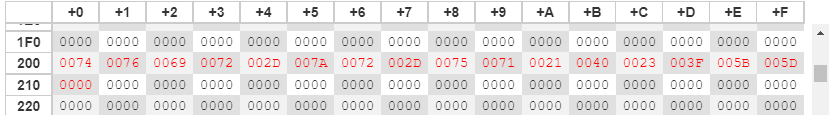
Test Case 2:

Input String: give-ME-HD!@#?[]

Output String: give-ME-HD!@#?[]  
 give-me-hd!@#?[]  
 tvir-zr-uq!@#?[]

****

**Figure 2.6.3 Convert to encode string.**

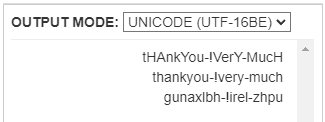
****

**Figure 2.6.4 Memory encode string**

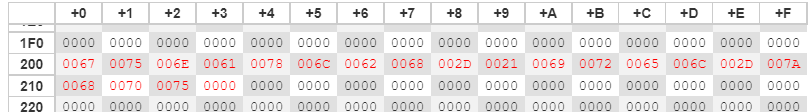
Test case 3:

Input String: tHAnkYou-!VerY-MucH

Output String: tHAnkYou-!VerY-MucH  
 thankyou-!very-much  
 gunaxlbh-!irel-zhpu

****

**Figure 2.6.5 Convert to encode string.**

****

**Figure 2.6.6 Memory encode string**