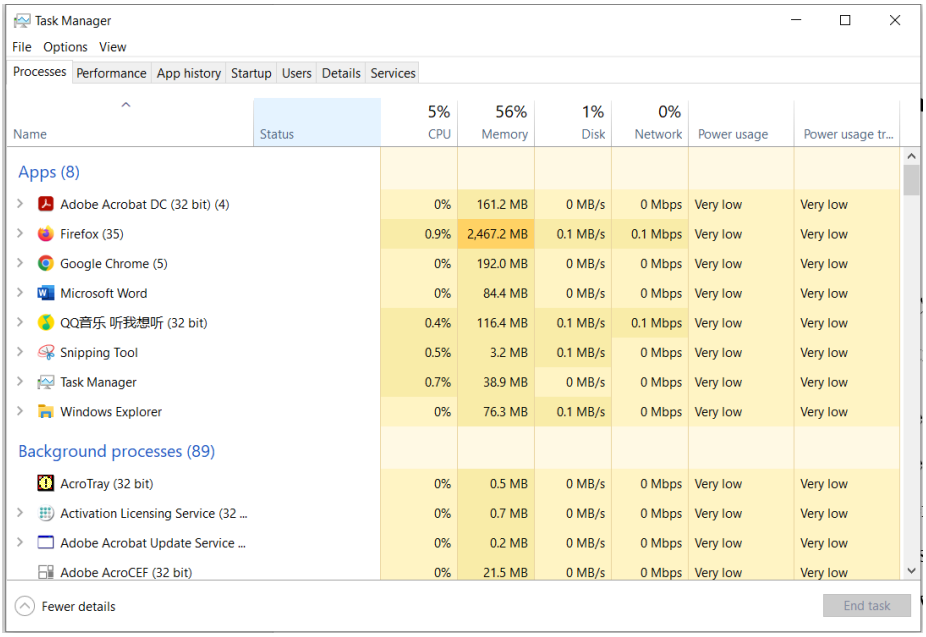
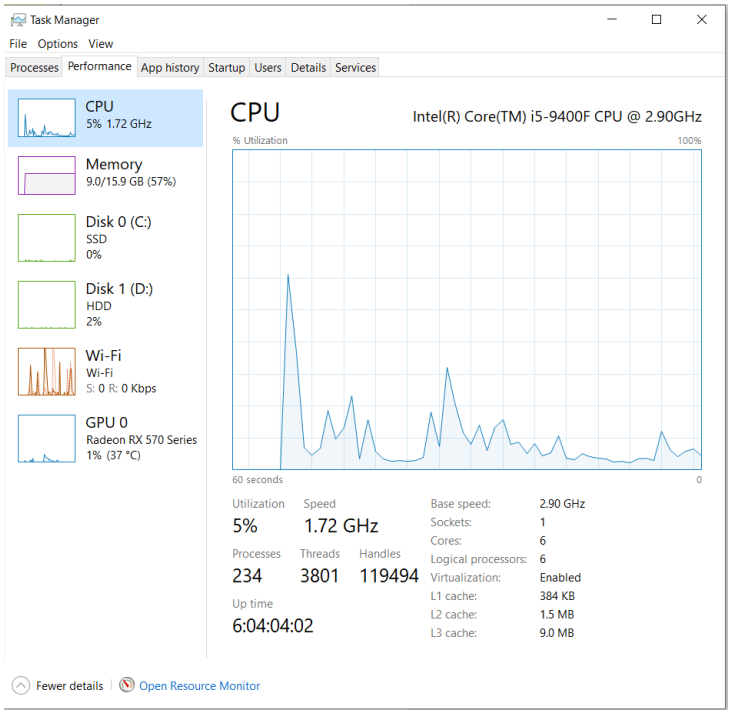
Part 1: Processes (589 words in total)

Task 1.1: (228 words)





In task manager, the first column is the CPU usage and the second is memory usage. The device is powered by an Intel i5 CPU sitting on 16-Gb memory space.

It was first noted that a lot more processes running in the system (background) as compared to the number of opened applications. More interesting, over 56% of the memory (>9 Gb) was

occupied. However, by adding up the memory of all processes in the task manager, the presented usage (~4Gb) is measurably less. This is explained that the task manager is a high-level overviewing platform that only shows the selected processes. Whereas, for example, the windows system itself takes a large portion of the memory while running.

Additionally, for a given process, its CPU usage and memory usage are not related. For example, internet explorer takes more than 2.4 Gb of memory while using only 0.6% of the CPU. This is because many websites are all opened simultaneously - all memory but need little computational power. On the contrary, as typing in Word is happening, the CPU usage spike up to 16% with only 136Mb of memory took – CPU usage with typing but little data being written. Lastly, the CPU usage fluctuates whereas memory usage is relatively stable for a process – when not actively being used it would not require CPU, but its memory space remains on RAM.

Task 1.2: (217 words)

Picked process name: COM surrogate – operated by dllhost.exe

From Task Manager Processes

Service Host: Local Service (Net...	0%	1.8 MB	0 MB/s	0 Mbps	Very low	Very low
COM Surrogate	0%	1.8 MB	0 MB/s	0 Mbps	Very low	Very low
Adobe Collaboration Synchroni	0%	1.8 MB	0 MB/s	0 Mbps	Very low	Very low

From Task Manager details – multiple running in dllhost.exe

dllhost.exe	5716	Running	SYSTEM	00	888 K	Not allowed
dllhost.exe	5480	Running	RLZ	00	1,872 K	Disabled
dllhost.exe	23148	Running	RLZ	00	856 K	Enabled

From online research on COM Surrogate, it is understood that the abbreviate – COM stands for Computer Object Management. Microsoft originally introduced this back in 1993 to assist/extend other applications with the capability of creating and operating “COM objects”. Accordingly, the COM was developed as a versatile tool that could run in/with different programming environments. One typical example of a COM object operation is the thumbnails in Windows file explorer (file manager). Upon opening a folder which contains media files (e.g., images or videos), the COM object running within the file manager processes these files and creates thumbnail images for display.

One issue with the original COM was that the COM runs within the ‘mother’ application as a single process. Subsequently, the failure of COM often causes the collapse of the main process. That is, a failed thumbnail generation crashes the entire application. To address this, Microsoft developed the COM Surrogate, which enabled the same functionality of COM object as its original form while allowing it to be run on a separate/independent process. This effectively protects the host application from this side extension. In a modern windows system, the COM surrogate is one essential process for computer operation.

Task 1.3: (144 words)

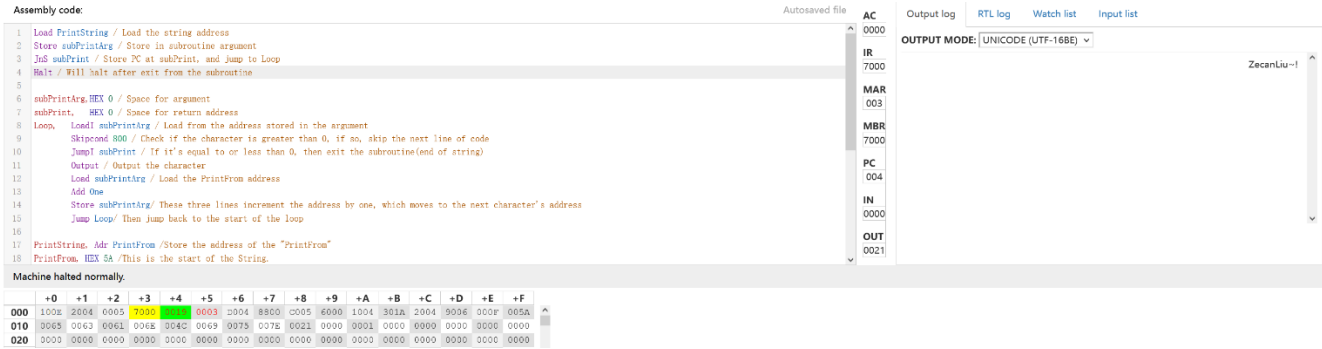
If each application manages the files itself, users must use different approaches to access the files between different applications. This will cause inefficiency and confusion. Operating system provides a unified way for the users and programmers to store and access data between different applications. Therefore, it allows the user to quickly locate and access the files. Sharing files between different users becomes easy as well.

Another unique function of the operating system is that it uses virtualisation to provide abstractions, which hides the complexity between consistent interfaces to users and programmers. This allows users to implement some functions without understanding the complex principles behind them.

As for security, it protects memory between processes by using virtual memory, which ensures the users’ confidential information in one process won’t be accessed by another process. It also protects the files of each user on the same computer.

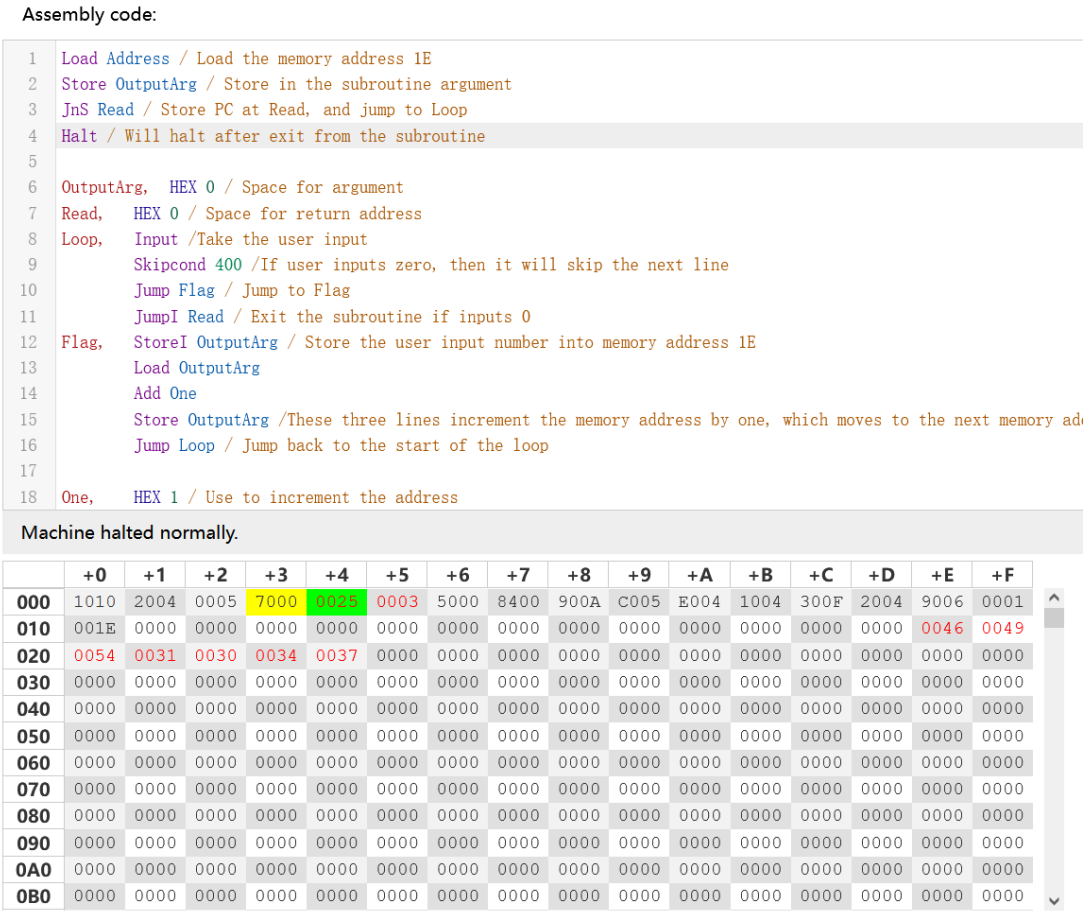
Task 2.3:

The memory addresses store my name starts from 000F.



Task 2.4:

The below screenshot is after inputting “FIT1047”.



Assembly code:

```
Load Address / Load the memory address 1E
Store OutputArg / Store in the subroutine argument
JnS Read / Store PC at Read, and jump to Loop
Halt / Will halt after exit from the subroutine

OutputArg,  HEX 0 / Space for argument
Read,    HEX 0 / Space for return address
Loop,    Input /Take the user input
         Skipcond 400 /If user inputs zero, then it will skip the next line
         Jump Flag / Jump to Flag
         JumpI Read / Exit the subroutine if inputs 0
Flag,    StoreI OutputArg / Store the user input number into memory address 1E
         Load OutputArg
         Add One
         Store OutputArg /These three lines increment the memory address by one, which moves to the next memory ad
         Jump Loop / Jump back to the start of the loop

One,     HEX 1 / Use to increment the address
```

Machine halted normally.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	1010	2004	0005	7000	0025	0003	5000	8400	900A	C005	E004	1004	300F	2004	9006	0001
010	001E	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0046	0049
020	0054	0031	0030	0034	0037	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

The below screenshot is after inputting “ZecanLiu~!”.

Assembly code:

```
Load Address / Load the memory address 1E
Store OutputArg / Store in the subroutine argument
JnS Read / Store PC at Read, and jump to Loop
Halt / Will halt after exit from the subroutine

OutputArg,  HEX 0 / Space for argument
Read,    HEX 0 / Space for return address
Loop,    Input /Take the user input
         Skipcond 400 /If user inputs zero, then it will skip the next line
         Jump Flag / Jump to Flag
         JumpI Read / Exit the subroutine if inputs 0
Flag,    StoreI OutputArg / Store the user input number into memory address 1E
         Load OutputArg
         Add One
         Store OutputArg /These three lines increment the memory address by one, which moves to the next memory address
         Jump Loop / Jump back to the start of the loop

One,     HEX 1 / Use to increment the address
```

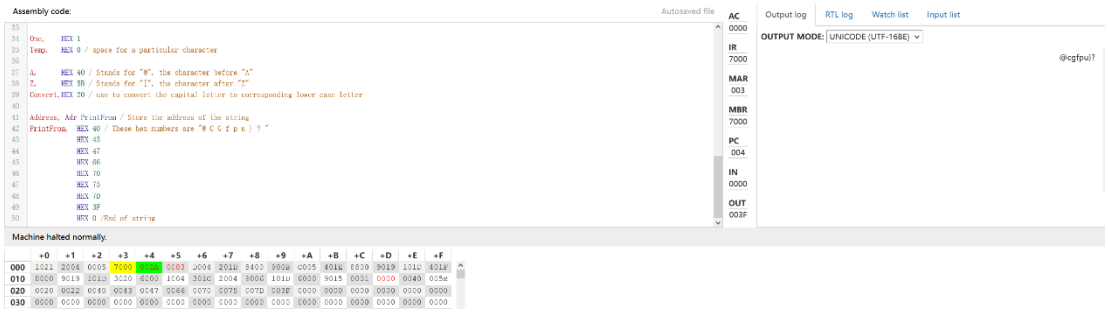
Machine halted normally.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	1010	2004	0005	7000	0028	0003	5000	8400	900A	C005	E004	1004	300F	2004	9006	0001
010	001E	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	005A	0065
020	0063	0061	006E	004C	0069	0075	007E	0021	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Task 2.5:

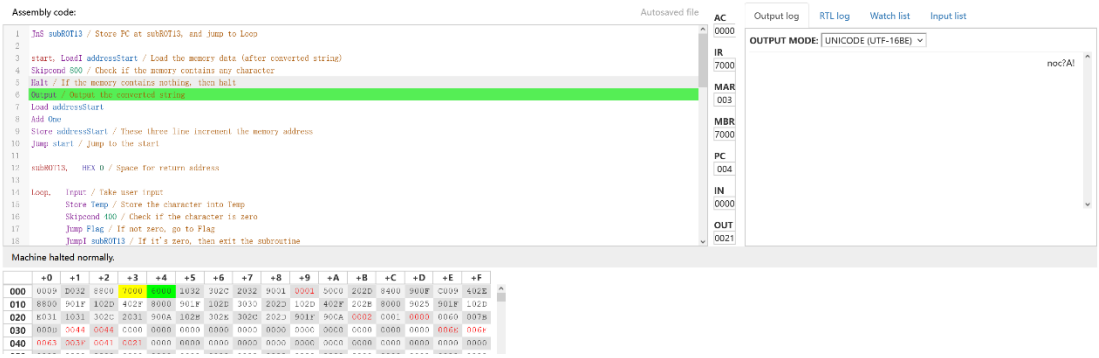
The below screenshot is for converting “5ABUNc5j”.

The below screenshot is for converting “@CGfpu}?”.

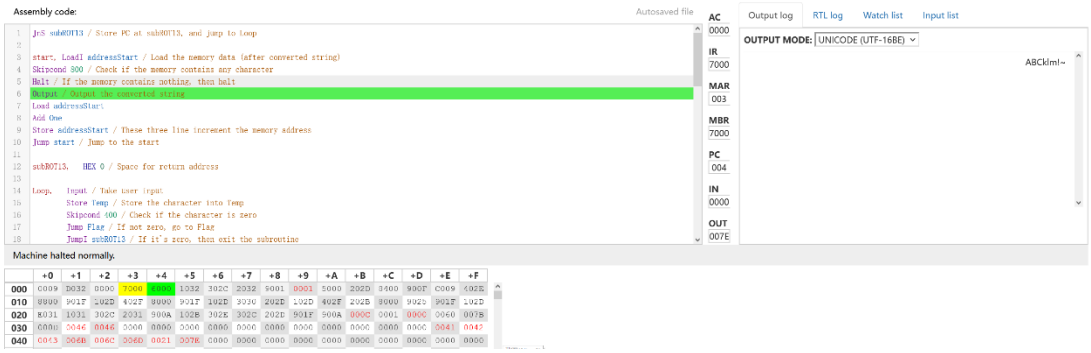


Task 2.6:

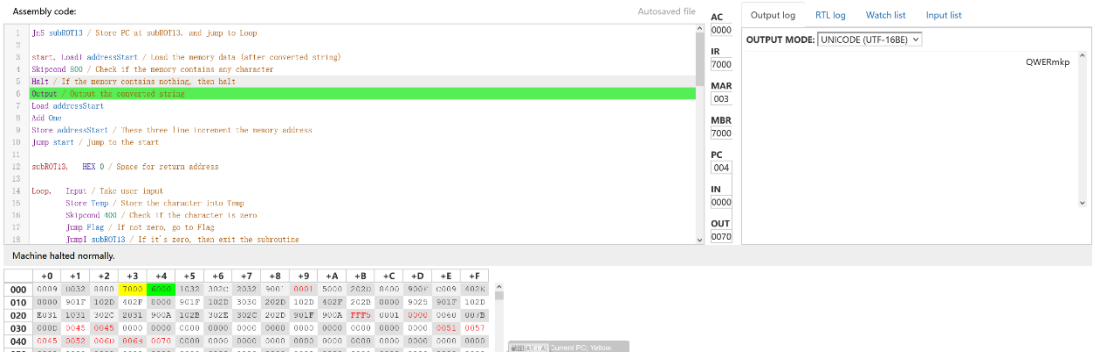
The below screenshot is for converting “abp?A!”.



The below screenshot is for converting “ABCxyz!~”.

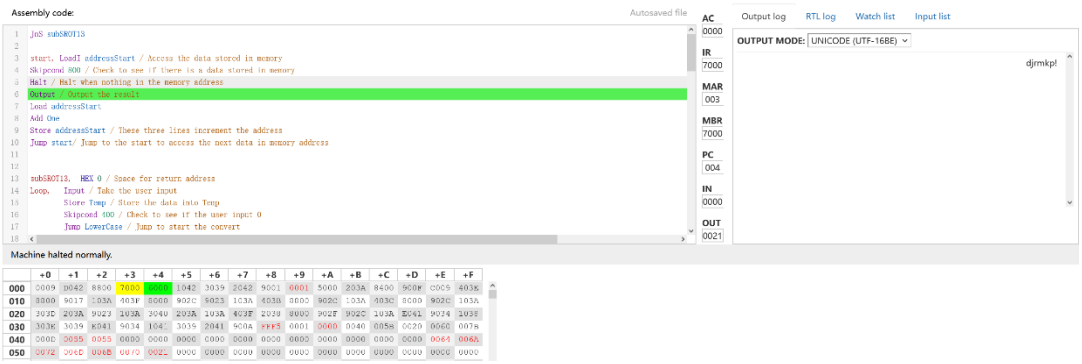


The below screenshot is for converting “QWERzxc”.

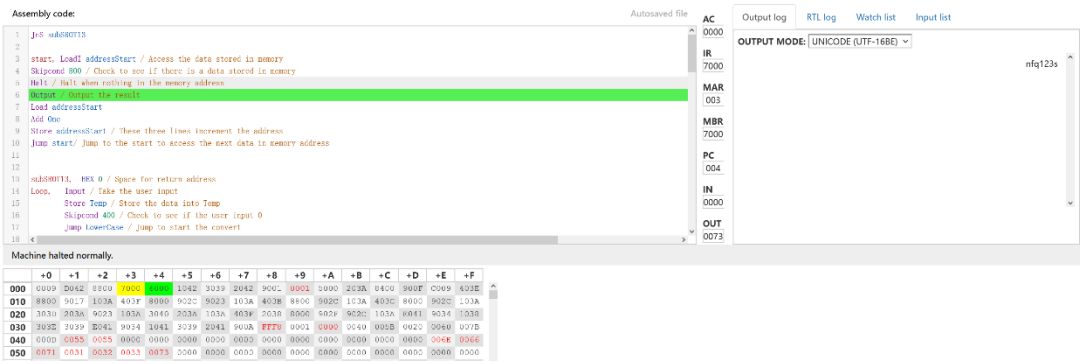


Task 2.7:

The below screenshot is for converting “QWEzxc!”.



The below screenshot is for converting “ASD123f”.



The below screenshot is for converting “abp?A!”.