2825-ASM 演讲稿

# **Reverse Loop**

Thank you. So I will start from the reverse loop.

First, compare ESI with EBX. The ESI stands for /

Extended (iks **dan** did) Source Index, and we use it as a counter.

Next, jump into write\_out / if ESI is Greater than / or Equal to / EBX. This is the condition to break the loop.

We already know that / EBX stores the length of the text, so the loop will stop / when the index —— which is ESI, / reach**es** the text length.

After that, we copy the length from EBX to ECX, – because / EBX is **in** the condition, so we don't want to modify it. / So, we use ECX in**st**ead.

The following part is the real function to reverse.

The first step / is to identify the character we want to **copy** in this loop.

Decrease ECX, – because we want to / move the pointer / for**ward** / one character.

Then, use / ECX minus ESI / to **skip** the character / that has already **been** copied.

The second step is to copy the character / from the input buffer / to AL.

AL refers to the lower eight bits of AX, and AX refers to the lower half / of EAX. It only has 1 byte; – we are using this because one character is one byte. The address of the input buffer / plus ECX / gives / the **address** of the character / we want to copy. Put that into AL.

The third step is to paste the character from AL / to the reverse variable. It’s similar to the previous step, -- reverse variable / plus the index / is the ad**dress** / of the current character.

Next step, increase the index and jump back to rev\_loop to repeat the process.

The C code on the right / **does** the same thing here.

# (NEXT SLIDE)

# **Output**

Here is the output. First, EAX 4, – which will call the system write.

Then, fill in the **pa‧ram‧e‧ter** **/pəˈramədər/** / for the system **write** function — select standard output, store the pointer, and store the length.

Then, use a system call to execute.

# (NEXT SLIDE)

# **Exit**

At the end, system call **one** / **ex‧its** (Egg sit s) the program / wi**th** return code 0, as we have mentioned that / XOR (ex-or) is faster than assigning (a 塞 ning) a number directly (呆 rec t ly), so we use XOR to return 0.

And that’s all. Thanks for listening.

# 

# 问答准备

**in‧ter‧rupt** (inter **r啊!**pt) :(

**hex‧a‧dec‧i‧mal** (hex A **de**si mel)

Interrupt hex eighty is to trigger a system call in Linux