# Flowchart - assembly-procedure

**Clarify the task:**
I needed to print uppercase English letters from A to Z, each followed by a newline character, directly on the terminal using assembly language.

**Plan data storage:**
I reserved a single byte in the .bss section to hold the current character being printed (char) and defined a newline byte (0xA) in the .data section for line breaks.

**Initialize the loop:**
Starting from ASCII 'A', I loaded the character into the cl register as the loop counter.

**Loop through characters:**
In each iteration, I compared cl with 'Z' + 1 to know when to stop.
Before calling the print procedure, I saved the current character with push cx to preserve it across the procedure call, then restored it after with pop cx.
After printing, I incremented cl to move to the next letter and looped again.

**Create a print procedure:**
The print_char procedure took the character in cl, stored it in memory ([char]) so I could pass its address to the Linux syscall.
Then I used the sys_write syscall twice—first to print the character, then to print the newline—ensuring each letter appeared on its own line.

**Graceful program termination:**
After printing all letters, I invoked the Linux exit syscall to terminate the program cleanly.

**Automate assembly and execution:**
To streamline testing, I wrote a bash script run.sh that accepts the assembly filename, assembles it with NASM, links it with ld, and runs the resulting executable—all with a single command.

**Set permissions and run:**
I gave execute permission to the script (chmod +x run.sh) and ran it with the command ./run.sh az, which successfully displayed the alphabet from A to Z on separate lines.

**Result:**
The program worked as expected—efficiently printing all uppercase letters with clear structure and reusable procedure design.