Avoiding Common Errors Committed During Programming

1. Size of the name given to .asm file – File name has to be less than 8 characters. So if you a name a file as myfirstprog.asm and try to assemble it the assembler will throw an error saying file not found.
    a. Keep the name of the file short and less than 8 characters
2. When saving a asm file saving it as a text document will name the file as myprog.asm.txt.
    a. Save the file using all file type option or better use notepad++ which allows you to save the file as an asm file. Download notepad++ from site [https://**notepad**-plus-plus.org/**download**/](https://notepad-plus-plus.org/download/)

3. Size of data
   (a) If you are working with bytes (8-bit data)
      I. Data should be stored as follows
         dat1 db 23h,56h,72h
      II. Data should be transferred from memory into an 8-bit register
         mov al,[si]
         using the following is incorrect
         mov ax,[si]
         mov eax,[si]
      III. Pointer increment or decrement should be done by one

         inc si /dec si

   (b) If you are working with 16-bit data
      IV. Data should be stored as follows
         dat1 dw 6723h,1A56h,72ACh
      V. Data should be transferred from memory into an 16-bit register
         mov ax,[si]
         using the following is incorrect
         mov al,[si]
         mov eax,[si]
      VI. Pointer increment or decrement should be done by two

         inc si

         inc s

(c) If you are working with 32-bit data

VII. Data should be stored as follows

dat1 dd 9178AA23h,0AABBCC56h, 1122334472h

VIII. Data should be transferred from memory into a 32-bit register

mov eax,[si]

using the following is incorrect

mov al,[si]

mov ax,[si]

IX. Pointer increment or decrement should be done by four

inc si

inc si

inc si

inc si

4. If you are using 16-bit registers as pointers you can use only si, di, bx and bp. When using bp only relative addressing is allowed. An offset should always be specified with bp

   mov bl,[bp+0]

   using register indirect addressing is incorrect

   mov bl,[bp]

   The assembler does not throw an error – because it automatically corrects it to *mov bl,[bp+0]*

5. When storing a string of alphanumeric characters – all operations will be in bytes only. Each character is represented by an 8-bit ASCII equivalent. So storage must be as follows

   str1 db "a,b,c,d"

   using the following is incorrect

   str1 dw "a,b,c,d"

   str1 dd "a,b,c,d"

   All rules related to byte transfer apply here.

6. EQU directive can be used only for declaring constants – it cannot be used for storing data in memory. So if while attempting to do

   cnt1 equ 5

   and then try to so something like

dec cnt1

system will throw an error.

So the correct method would be

cnt1 db 5

dec cnt1

7. The size of source and destination must be the same
   add si,cl is incorrect – if contents of cl needs to be added with si – use the following method
   mov ch,0
   add si,ax

8. When using .model tiny use ml for assembly and linking – masm followed by link should not be used.

9. The code segment starts with .startup and ends with .exit. If .startup is missing – segment registers will not be initialized.

10. For executing the complete code if ip is pointing to 100 – execute until the address where mov ah,4ch followed by int 21h is present.
    For e.g.
    If
    cs:112 mov ah,4ch
    cs:114 int 21h
    execute up to 112 using either *g 112 (if ip is at 100) or using g =100 112 (if ip is not at 100)*
    do not use *g* as int 21h – causes the segment values to change.

11. When using string instructions the source address is always a combo of ds:si and destination is always es:di.

12. rep/repe/repne prefix can be used only with string instructions.

13. macro is defined before .code and .startup

    For e.g.

```
capson      macro
            cmp         al,'a'
             jl         x2
            cmp          al,'z'
            jg          x2
            sub         al,20h
x2:
            endm

.code
.startup
```

14. When using subroutines, software int instructions - stack must always be initialized and hence also the stackpointer

    For e.g.

```
st1     dw      10 dup(?)

st2     dw       ?
```

    In the code

```
lea     sp,st2
```

    [Note: st2 offset must be loaded into sp not st1 as stack moves backwards]

15. Stack is usually initialized as word not bytes - as stack transfer is usually in words.
16. Data entered from keyboard/ read or written into files all are in ASCII format - so if you want display 6 or store 6 in file the value you have to give is 36. If you enter 3 from keyboard or read 3 from file value will be ASCII so you will have 33 in register or memory
17. DAA can be used only after addition with the accumulator - the rules of programming will have to be followed.