

Display 8bit number

Design

8 bit number $\rightarrow 4$
ASC -II Value $\rightarrow 34$
$$\begin{array}{r} 34 \\ - 30 \\ \hline 4 \end{array}$$

BH \leftarrow 04H
BH \leftarrow 40H

8 bit number $\rightarrow 3$
ASC -II Value $\rightarrow 33$ AL
$$\begin{array}{r} 33 \text{ AL} \\ - 30 \\ \hline 3 \text{ AL} \end{array}$$

Add AL & BH $\rightarrow 40 + 3$
BH = 43H

Num \leftarrow 43H
BH \leftarrow 43
AND 43
OF 0
$$\begin{array}{r} 43 \\ 0F0 \\ \hline 40H \end{array}$$

SHR BH \rightarrow 04H

ADD 30
+ 04
$$\begin{array}{r} 34 \\ + 04 \\ \hline 34 \rightarrow DL \end{array}$$

BH \leftarrow Num
BH \leftarrow 43

43
OF AND
$$\begin{array}{r} 43 \\ 0F \text{ AND} \\ \hline 03 \end{array}$$

BH \rightarrow 03H

ADD 30
+ 03
$$\begin{array}{r} 33H \\ + 03 \\ \hline 33H \rightarrow DL \end{array}$$

Addition/Subtraction

Application: Use of Macros and procedure in the Assembly Language programming modular program.

Design: -

Structure of Macro

DISP - macro xx → parameters
└──┬──┘
name of macro. assembler directive.

Disp macro xx
⋮ } set of instruction
END M → end of macro

Structure of procedure

Disp1 Proc xx → parameters
└──┬──┘
name of procedure assembler directive

Disp1 PROC
⋮

⋮ } Set of instruction

Ret → Return the value.

DISP 1 END P → END of procedure

Definition of macros:-

The assembly macros which stands for macro instruction is a programmable pattern which translates a certain sequence of input into a present sequence of output.

Definition of procedure:-

A procedure is a group of instruction that usually perform one task. It is reusable section of a software program which is stored in memory once but can be used as often as necessary.

Hex to bcd

HEX to BCD

Initially Counter = 0

hex = 0AC

mov ax, hex $ax \leftarrow \text{hex}$

$ax \leftarrow 0ACH$

mov bx, 000AH $bx \leftarrow 000A$

counter = 1

div bx

$AX \leftarrow AX / BX$ $000ACH / 000A \leftarrow 0011$

$dx \leftarrow AX \% BX$ $000ACH \% 000A \leftarrow 002$

Quotient in AX.

Remainder in dx.

cmp dx, 0 $dst \rightarrow \text{Source}$.

0011, 0000 $ZF=0, CF=0, SF=0$

$dx = 00$

je exit If $ZF=1$ then exit

counter = 2

Since $ZF=0$; therefore again
div bx

$AX \leftarrow AX / BX = 0011 / 000A = 01$

$dx \leftarrow dx / BX = 0011 / 000A = 07$

$\therefore \text{CMP } dx, 0$.

$0001 > 000$

$CF=0, SF=0, ZF=0$

$dx = 0$

counter = 3

div bx

$ax \leftarrow ax / bx \leftarrow 0001 / 000A \leftarrow 000$

$dx \leftarrow ax \% bx \leftarrow 0001 \% 000A \leftarrow 0001$

cmp ax = 0

Since $dst = \text{Source}$.

$\therefore ZF=1, CF=0, SF=0$

\therefore Now $CL=3, CH=00$; CL store the
counter value.

\therefore Now Loop LI

$cx = cx - 1$

$cx = 3 - 1$

$= 2 - 1$

$2 \neq 0$

$cx \neq 0$

Similarly;

$cx \neq 0$

Loop end

Negative No

How to declare Uninitialized Array:

Syntax:

(Example) `a DB 7 DUP(0)`

where: 'a' → Name of array given.

DB → It is written on the space of datatype which represents which type of data we are considering
eg DB → 8 bits DW → 16 bits of data

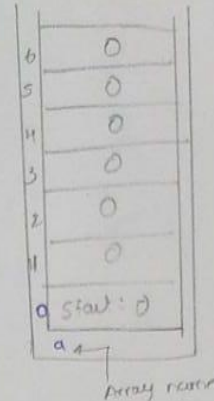
7 → It represents Size of the array assign.

dup(0) → It used for uninitialized array declaration.
It means assign 0 to the space mentioned for array.

Syntax:

array_name Data_type Size_of_array DUP(0)

* How Array is created for uninitialized declaration
`a DB 7 DUP(0)`



* How to Declare Initialized Array.

Syntax:

Array_name Data_type Array_Elements

Array_name: Name assigned to array.

Data_type: The data type of the array elements.

Array_Elements: The value of the array which is to be stored in Array_name

* How array is stored

96H
94H
92H
90H
8EH
8CH
start: 25H
arr

Example: `arr DB 25H, 41H, 36H, 38H, 34H, 91H, 96H`

* To Check if a number is negative

Method 1: Shift Left by 1

If after performing Left shift operation the carry flag (CF) is set to 1 the number is negative, else if CF=0 then number is positive.

Eg 1 $\begin{matrix} Cy & 0010101 & shl & by & 1 \\ 0 & 10010101 & & & \end{matrix}$ $\begin{matrix} Cy & 00101010 \\ 1 & 00101010 \end{matrix} \Rightarrow \text{Negative}$

Eg 2 $\begin{matrix} Cy & 01100110 & shl & by & 1 \\ 0 & 01100110 & & & \end{matrix}$ $\begin{matrix} Cy & 11001100 \\ 0 & 11001100 \end{matrix} \Rightarrow \text{Positive}$

Method 2: AND with 80H

when we perform AND operation with 80H to any number and if zero flag is set to 1 (ZF=1) then number is positive else for ZF=0 the number is negative

Eg $\begin{matrix} 10101100 \\ 10000000 \\ \hline 10000000 \end{matrix}$

ZF=0

∴ Negative number

$\begin{matrix} 00111111 \\ 10000000 \\ \hline 00000000 \end{matrix}$

ZF=1

∴ Positive Number.

String operation

1. How to accept string from user:

function: `MOV AH, 0AH`

`LEA DX, str-variable`

`INT 21H`

2) How to declare string in data structure

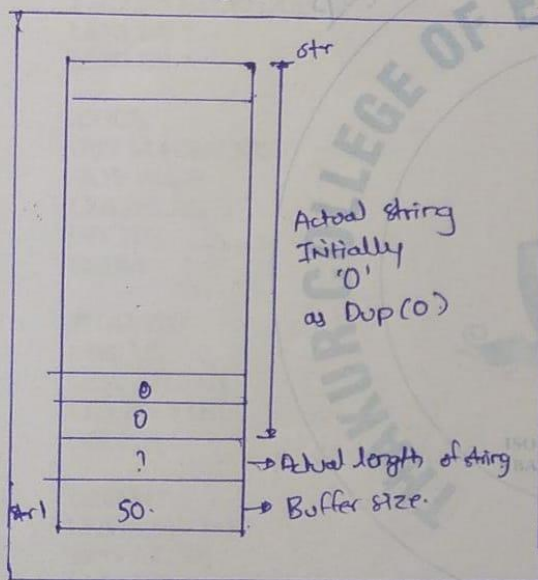
format / syntax

Name-str DataType Buffer, ?, Actualsize Dup(0)

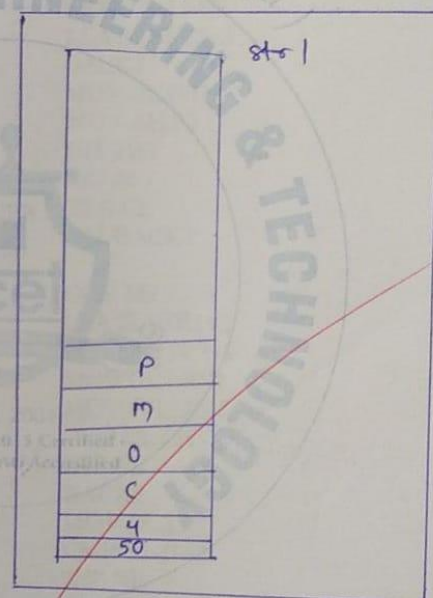
Ex:- `str1 DB 50, ?, 50 Dup(0)`

Result and Discussion

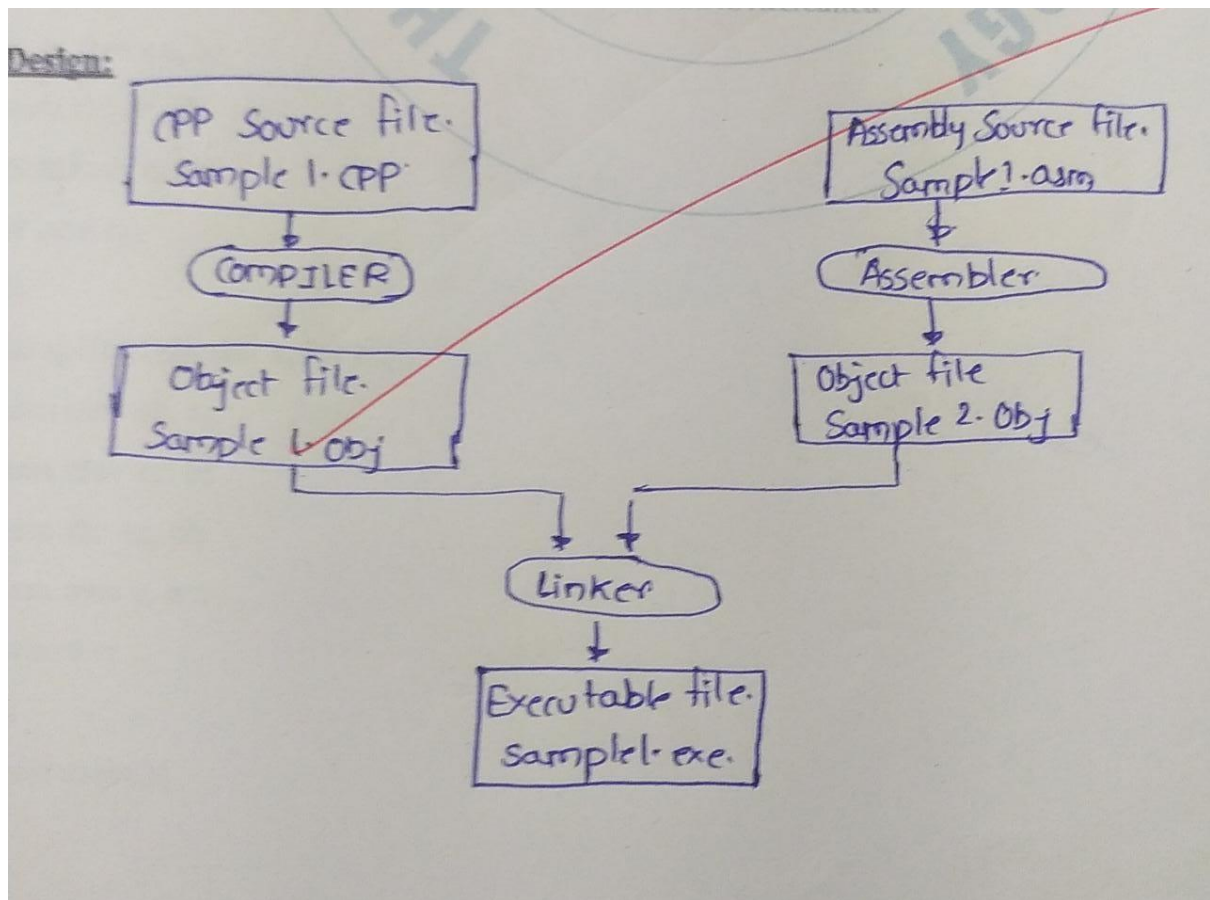
Data structure:



Input: comp



Mix mode



Mouse

Design

Interrupt to access the mouse driver

INT 33H

TCET

1. AX = 0000 → checks status of INT 33H mouse driver
Returns for:

AX ≠ 0; mouse driver present
AX = 0; mouse driver not present.

TCET

2. AX = 0001; To check mouse.

INT 33H cursor/show the mouse cursor

3. AX = 0004H; To initialize the mouse cursor position

; This function will store:

; CX = X - co-ordinate

; DX = Y - co-ordinate

INT 33H.

4. AX = 0007H

; It is used to set horizontal limit

; CX = min X-co-ordinate

; DX = max X-co-ordinate

INT 33H.

5. AX = 0008H

; It is used to set the vertical limit

; CX = min Y-co-ordinate

; DX = max Y-co-ordinate

INT 33H

6. AX = 0003H

; used to get mouse button status

INT 33H

Returns →

BX = 1 (Left button clicked)

BX = 0 (Center button clicked)

BX = 2 (Right button clicked)

- 7) Text mode (To set Text

AX = 0000H mode)

INT 10H

- 8) To set graphics mode.

AX = 0011H

INT 10H

- 9) To display pixel

AH = 0CH

INT 10H.