CSE 331/EEE 332 (Microprocessor Interfacing & Embedded System Lab)

Lab 04   : Library of common functions - emu8086.inc
Lab Officer   : Azmary Zannat Aurin

Loop

The 'Loop' instruction provides a simple way to repeat a block of statements a specific number of times. To do this, first we have to put a value in CX (count resister) which will track the number of times the block of instructions will execute.

Instruction	Meaning	Examples
LOOP	Keep executing this instruction including the instruction of the target label until the CX register becomes zero.	MOV CX, 2345H HERE: NOP LOOP HERE

Example:

C code
int j;

j = 0;

while (j < 10)
{   j++
;
}

Assembly code

MOV AL, 0
MOV CX, 10

INCREMENT:

ADD AL, 1

LOOP INCREMENT

Macro

- A macro is a named block of assembly language statements.
- Once defined, it can be invoked (called) one or more times.
- During the assembler's preprocessing step, each macro call is expanded into a copy of the macro.
- The expanded code is passed to the assembly step, where it is checked for correctness

Defining macros:
- A macro must be defined before it can be used.
- Parameters are optional.
- Each parameter follows the rules for identifiers. It is a string that is assigned a value when the macro is invoked.

Macroname MACRO [parameter-1, parameter-2, ...]

  statement-list

ENDM

Example:

INCLUDE 'EMU8086.INC'

.MODEL SMALL .STACK 100H

.DATA

.CODE

ADD_TWO MACRO R1

  MOV AX, R1

ENDM ADD_TWO

MAIN PROC

  MOV CX, 5   ADD_TWO

CX ENDP MAIN

END MAIN

Library of common functions - emu8086.inc

To make programming easier there are some common functions that can be included in your program. To make your program use functions defined in other file you should use the INCLUDE directive followed by a file name. Compiler automatically searches for the file in the same folder where the source file is located, and if it cannot find the file there it searches in Inc folder.

To use any of the functions in emu8086.inc you should have the following line in the beginning of your source file:

include 'emu8086.inc'

emu8086.inc defines the following **macros**:
- **PUTC char** - macro with 1 parameter, prints out an ASCII char at current cursor position.
- **GOTOXY col, row** - macro with 2 parameters, sets cursor position.
- **PRINT string** - macro with 1 parameter, prints out a string.
- **PRINTN string** - macro with 1 parameter, prints out a string. The same as PRINT but automatically adds "carriage return" at the end of the string.
- **CURSOROFF** - turns off the text cursor.
- **CURSORON** - turns on the text cursor.

Example 1

```
INCLUDE 'EMU8086.INC'

.MODEL SMALL .STACK 100H

.DATA

.CODE

MAIN PROC

    PRINT 'HELLO WORLD!'

    GOTOXY 10, 5

    PUTC 65        ; 65 - IS AN ASCII CODE FOR 'A'    PUTC 'B'

ENDP MAIN

END MAIN
```
emu8086.inc also defines the following procedures:

- **PRINT_STRING** - procedure to print a null terminated string at current cursor position, receives address of string in DS:SI register. To use it declare: DEFINE_PRINT_STRING before END directive.

- **PTHIS** - procedure to print a null terminated string at current cursor position (just as PRINT_STRING), but receives address of string from Stack. The ZERO TERMINATED string should be defined just after the CALL instruction. For example:

  CALL PTHIS db
  'Hello World!', 0

  To use it declare: DEFINE_PTHIS before END directive.

- **GET_STRING** - procedure to get a null terminated string from a user, the received string is written to buffer at DS:DI, buffer size should be in DX. Procedure stops the input when 'Enter' is pressed. To use it declare: DEFINE_GET_STRING before END directive.

- **CLEAR_SCREEN** - procedure to clear the screen, (done by scrolling entire screen window), and set cursor position to top of it. To use it declare: DEFINE_CLEAR_SCREEN before END directive.

- **SCAN_NUM** - procedure that gets the multi-digit SIGNED number from the keyboard, and stores the result in CX register. To use it declare: DEFINE_SCAN_NUM before END directive.

- **PRINT_NUM** - procedure that prints a signed number in AX register. To use it declare: DEFINE_PRINT_NUM and DEFINE_PRINT_NUM_UNS before END directive.

- **PRINT_NUM_UNS** - procedure that prints out an unsigned number in AX register. To use it declare: DEFINE_PRINT_NUM_UNS before END directive.

To use any of the above procedures you should first declare the function in the bottom of your file (but before the **END** directive), and then use **CALL** instruction followed by a procedure name. For example:

INCLUDE 'EMU8086.INC'

.MODEL SMALL .STACK 100H

.DATA

MSG1 DB 'ENTER THE NUMBER: ', 0

```
.CODE

MAIN PROC

    MOV AX, @DATA     MOV DS, AX

    LEA    SI, MSG1      ; ASK FOR THE NUMBER     CALL PRINT_STRING ;
    CALL SCAN_NUM       ; GET NUMBER IN CX.
    MOV   AX, CX         ; COPY THE NUMBER TO AX.

    ; PRINT THE FOLLOWING STRING:
    CALL PTHIS      DB 13, 10, 'YOU HAVE ENTERED: ', 0

    CALL PRINT_NUM        ; PRINT NUMBER IN AX.
    DEFINE_SCAN_NUM    DEFINE_PRINT_STRING    DEFINE_PRINT_NUM
    DEFINE_PRINT_NUM_UNS ; REQUIRED FOR PRINT_NUM.
    DEFINE_PTHIS

ENDP MAIN

END MAIN
```

Task 1

Take an input and print it in the console.

Task 2

Take two integer values from the user. And print their sum value.

Task 3

Create an array of five elements and search for an element. If the element exists in the array print 'Value found' otherwise 'Value not found'

Task 4
A palindrome is a word that is same when read from both ends. Example: 'RACECAR'. Write a program that will check if a string is palindrome or not. The output may be a 'Yes' or a 'No'.

Example:
racecar yes
hellyeah
no

Task 5
Create an array of size 5 with user input. Print the largest value.