Computer Organization and Assembly Language

Lab Manual (Lab 02)

Topic: Program segment directives, Memory model code segment, Data segment, Stack segment, Data variable, Program segment structure, the template of an assembly program to print a string.

Course Instructor: Saima Shaheen

Lab Instructor: Saima Shaheen

Session: Spring 2023

School of Systems and Technology

UMT, Lahore, Pakistan

Objectives: Displaying a sting using assembly language program.

Some concepts and the template of an assembly program:

The following program displays the string “Hello, World! “ on the screen.

```
; Assembly Program: Hello, World!

.8086

.model small

.stack 100h

.data

message db "Hello, world!",0dh,0ah,'$'          ; define byte in a memory (having name message), a string of "Hello world), dollar sign represent end of string

.code

main proc ; main procedure, code segments starts from main procedure

mov ax,@data ;loading starting address of data segment in AX

mov ds,ax    ;Initialize DS

mov ah,9h ; DOS Function call to print a message

mov dx,offset message

int 21h

mov ax,4C00h ; DOS Function call to exit (Preparing to exit)

int 21h ; interrupt command (used in combination with ah, ****h            )
                                                                    command

main endp ;procedure end

end main  ; code segment end
```

Results and Observations

Memory Model for 8086 ISA

Program Segment Directives

The key simplified segment directives are **STACK_SEG**, **CODE_SEG**, **DATA_SEG**, **.MODEL**.

Memory Models

The **.MODEL** directive specifies the memory model for an assembler module that uses the simplified segment directives. The .MODEL directive must precede .CODE, .DATA, and .STACK. The format of the **.**MODEL directive is:

.MODEL memory model

The memory model can be **TINY**, **SMALL**, **COMPACT**, **MEDIUM**, **LARGE and HUGE**.

TINY	One segment. Thus both program code and data together must fit within the same 64 Kb segment. Both code and data are near.
SMALL	Program code must fit within a single 64 Kb segment, and data must fit within a *separate* 64 Kb segment. Both code and data are near.
MEDIUM	More than one code-segment. One data-segment. Thus code may be greater than 64K.
COMPACT	One code-segment. More than one data-segment. Thus data may be greater than 64K.
LARGE	More than one code-segment. More than one data-segment. No array larger than 64K. Thus both code and data may be greater than 64K.
HUGE	More than one code-segment. More than one data-segment. Arrays may be larger than 64K. Thus both code and data may be greater than 64K

All program models but TINY result in the creation of **exe-format** programs. The TINY model creates **com-format** programs.

Stack Segment

STACK_SEG defines the stack segment and controls the size of the stack. For example,

STACK_SEG segment STACK 200h

Defines stack 200h (512) bytes long.

.stack 100h is used so far (see above code)

Code Segment

CODE_SEG marks the start of your program's code segment. It contains executable instruction.

Main PROC

; Here the lines of executable instructions.

Main endp

CODE_SEG ends

.code is used so far (see above code)

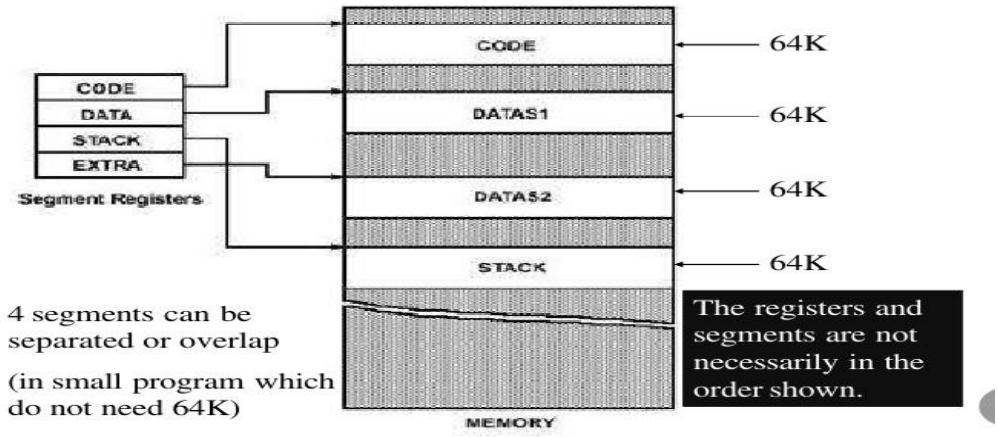
Data Segments

DATA_SEG marks the start of your data segment. You should place your memory variables in this segment. For example,

```
DATA_SEG segment ‘data’
Coal db "Welcome to coal"
DATA_SEG ends
```

.data is used so far (see above code)

- CS: points to the beginning of the code segment
- DS: points to the beginning of the data segment
- SS: points to the beginning of the stack segment



The Intel 8086 Registers

Accumulator register (AX) Accumulator can be used for Arithmetic, I/O operations and string manipulation.
Base register (BX) BX register usually contains a data pointer used for based, based indexed or register indirect addressing.

Count register (CX) Count register can be used as a counter in string manipulation and shift/rotate instructions.
Data register (DX) Data register can be used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.
The following registers are both general and index registers:
Stack Pointer (SP) is a 16-bit register pointing to program stack.
Base Pointer (BP) is a 16-bit register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing.
Source Index (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data addresses in string manipulation instructions.
Destination Index (DI) is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data addresses in string manipulation instructions.
Other registers:
Instruction Pointer (IP) is a 16-bit register.
Flag register is a 16-bit register.

DATA VARIABLES

symbolic addresses of data items (offsets in the data segment)
Defined by directives DB, DW, DD, DQ
Format for data definition

[name]	Dx	expression
Coal	DB	"this is my first program"

Directive Dx:

- determines the variable type (according to the letter x)
- allocates the space in memory (one or more bytes)
- Initializes the contents of the memory locations (does not Initialize, if the expression is?)

Expression:
can be uninitialized : ?
Can be assigned a constant: such as 25, 21.
Example:
DATAZ DB 21, 2

Directive	Allocated memory size in bytes	Variable type	Variable may contain
DB	**1**	**BYTE**	**Signed integer in the range <-128; 127> Unsigned integer in the range <0; 255> Character**
DW	**2**	**WORD**	**Signed integer in the range <-32 768; 32 767> Unsigned integer in the range <0; 65 535> 16-bit offset**
DD	**4**	**Double Word**	**Signed integer Unsigned integer Single precision floating point number in the range about ±10^38**

			Far pointer in 16-bit mode, i.e. address in the segment: offset form 32-bit offset
DQ	**8**	**Quad Word**	**Signed integer Unsigned integer Double precision floating point number in the range about ±10^308**

Comments

- anything following **;** on a line
- **;** This stuff would be considered a comment

Program Segment Structure

- Select a memory model
- Define the stack size
- Define data segment
- Declare variables
- Define code segment
- Write code
- organize into procedure
- Mark the end of the source file

LAB TASKS

Task # 1:

Write an assembly language program using 8086 syntax.

1. Print your Name
2. Print your Registration ID
3. Print your section
4. Output in following format:

Name: xyz

ID: 1234

Section: B