NAME	SAEEDA NAZ
ID	14332
Semester	5th
teacher	Sir Ameen
Assignment No	(3.)

Assignment No 3.

1) Using the value -35 write it an integer literal in decimal, hexadecimal, octal and binary formats that are consistent with MASM Syntax.

Ans :- -35d, 0Dh, 3350, 11011101b

2) Write the real number -6.2×10^4 as a real number literal using MASM Syntax.

Ans :- -62E+04

3) which Statement halts the assembly language program?

Ans :- The exit Statement (indirectly) call a predefined MS-window function that halts the program. The ENDP direct marks the end of the main procedure. The END main direct. movie the last line of the program to be assembled. DATA Type Essential characteristics. Size in bit 8, 16, 32, 48, 64 and 80.

4) what is a calling convention, and how is it used in assembly language declaration.

Ans :- A calling convertion determines how parameters are passed to subroutines

and how the stack is restored after the subroutine call.

5) Why might you use a symbolic constant rather than an integer literal in your code?

Ans :- An integer literal, such as 48, has no direct meaning to someone reading the programs sources code instead, a symbolic constant such as STUDENT_COUNT can be assignment an integer value and is self-documenting.

6) How is a source file different from a listing file?

Ans A source file is given as input to the assembler. A listing file has additing text that will not assemble. it is a file that is created by the assembler and it is optionally generated.

7) How are data label and code label different?

Ans :- DATA labels exist in the data segment as variable offsets. code labels are in the code segment and are offsets for transfer of control instructions. A code label is following by a colon but a data label does not with a colon.

(8) what type of files are produced by the assembler and linker?

Ans The main output produced by assembler an input assembly language source file is the translation of the file into an object file in (ELF). ELF files produced by the assembler are relocatable files that hold code and/or data. They are input files for the linker.

9) which Operating System component reads and executes programs?

Ans : A monolithic kerned runs all the operating System instruction in the same address space for speed. A microkerned run most processes in user space for modularity. This central component of a computer system is responsible for running or executing program.

OR

Ans The Loader.

10) Declare an unsigned 16-bit integer variable named warry that uses three initializers.

Ans : WArray WORD 10, 20, 30.

11) Declare a string variable containing that name of your favorite color. Initialize it as a null-terminated string.

Ans :- my color BYTE "blue", 0.

12) Write a statement that causes the assembler to calculate the number of bytes in the following array, and assign the value to a symbolic constant named ArraySize.

myArray DWORD 20 DUP (?)

Ans ArraySize = ($ - myArray)

13) Show how to calculate the number of element in the following array, and assign the value to a symbolic constant named ArraySize :

myArray DWORD 30 DUP (?)

Ans :- ArraySize = ($-myArray) / TYPE DWORD

14) Create an uninitialized data declaration for a 8-bit, 16 bit and 32 bit unsigned and signed integer.

- 8 signed - Var 2 SBYTE
- 16-bit signed integer.
- Var 1 SWORD
- 8-bit unsigned integer.
- Var 2 BYTE
- 32 bit signed integer.
- SDWORD.

15) Create a definition for a doubleword that Stored in memory in little endian format.

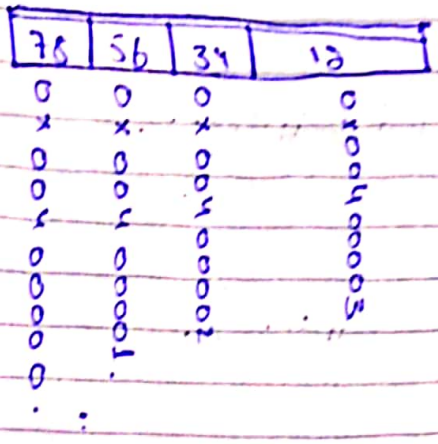
Little Endian :-

The least Significant byte (the "little end") of the data is placed at the byte with the lowest address. The rest of the data is placed in order in the next bytes in memory.

- In these definition, the data a 32-bit patterns in regarded as a 32-bit

unsigned integer. The "most significant" byte is the one for the largest power of two: $2^{31} \ldots 2^{24}$. The smallest power of two: $2^{7} \ldots 2^{0}$.

For example say that the 32-bit pattern 0x12345678 is stored at address 0x00400000



(16) Show the order of individual bytes in memory (lowest to highest) for the following doubleword.

Variable : val1 DWORD
87654321h

True or false : The following is a valid data definition of statement.

Var1 BYTE 0Ah, 255

Write an assembly program to compute that following expression.

AL = BL + val1

val1 is an 8-bit variable.

(7) Differeniate b/w equal sign directive and EQU directive.

Ans :- Equal-sign Directive :

The equal-sign directive associates a symbol name with an integer expression

This syntax is

name = Expression

- Expression is a 32-bit integer
- May be redefined.
- Name is called a symbolic constant

EQU - directive .

- Define a symbol as either an integer or text expression

- connot be redefined.

(18) Name the four basic part of an assembly language instruction.

Ans :- [label :] mnemonic [operands] [; comment].

19) Show an example of a block comment?

Ans :- COMMENT !
First line comment.
Second line comment.

20) You do not

20) Why it is not good idea to use numeric addresses when writing instructions that access variables?

Ans :- You do not use numeric addresses (offsets) for variables because the addresses would change if new variables were inserted before the existing ones.

9

21) What type of argument must be passed to the Exit Process procedure?

Ans :- An integer, Preferably 0.

22) Create a single integer expression that use all the operators. Calculate the value of the expression.

Ans :-

(5+1)/(-2+3)*2 mod 5 = 2

23) What type of arguments must be passed to the Exit process procedure?

Ans :- model small
386
Stack 100h

- data
  Sunday = 0
  Monday = 1
  Tuesday = 2
  wednesday = 3
  Thursday = 4

Friday = 5
Saturday = 6

Days DB Sunday, Monday, Tuesday, wednesday, thursday, Friday, Saturday.

. code
main:

```
mov ax, @data
mov ds, ax
; Just for test ; print the first value.

mov ah, 09h
mov dl, days
add dl, 30h
int 21h

mov ah, 4ch
int 21h

end main
```

24) Declare an array of 120 uninitialized unsigned double word value.

Ans :- my Array DWORD 120 (?)

35) Declare an array of 120 uninitialized unsigned double word X

35) Declare an array of byte and initialize it to the first 5 letter of the alphabets.

Ans :- my Array BYTE 'A', 'B', 'C', 'D', 'E'.

36) Find out if you can declare a variable of type DWORD and assign it a negative value.

Ans :- val1 DWORD 12345678h ; unsigned
val3 DWORD 30 DUP (?) ; unsigned.

37) Discuss the following MASK directives.

Ans :- • Stack directive :-

The Stack directive tell how many bytes of memory to reserve for the runtime stacks. 4096 happen to correspond to the size

of a memory page in this processor system for managing memory.

* • MODEL :-

This tells the assembler which memory model to use. In 32-bit Programs, we use the flat memory model which is associated with the processor's protected mode.

* • 386 :-

```
model flat.
.stack 4096
Exit process PROTO
dwExit code : DWORD.
```

The .386 directive identifies it as a 32-bit Program. Line2 uses the flat memory model, and windows requires the Model convention to be used.

Line 3 set aside 4096 bytes of storage.
Line 4 declares a photo type for the Exit process function.

* • CODE :-

```
.code
main PROC.
```

it is the beginning of the code area.

(4)

of the program (meaning & what's afterward is usually the main() procedure).

* • DATA :-
  • The DATA directive creates a near data segment.
• This DATA segment contain the frequently used data for your program.
• DATA segment can occupy
  ★ up to 64K in MS-DOS
  ★ or up to 512 mega byte under flat model in window NT.

* • photo :-
The ~~prot~~ PROTO directive's by using the INVOKE directive.

Syntax :-
label PROTO [[distance]] [[language-type]]
[[parameter]] : tag ...]

Parameter :-
distance (32-bit MASM only).
(optional) used in 16-bit memory models to override the default and indicate NEAR or FAR calls.

* PROC :-
Mask start and end of a Produce procedure block called label.

The statement in the block can be calle with the call instruction or INVOKE directives.

Syntax :-
* label PROC [distance] [language-type [PUBLIC | PRIVATE | EXPORT]]
* [[< protoguearg >]].
* [[USES regist] [parameter [[: tag]]...]].
* [[FRAME [[: ehandler - address]]]]

Statement
label End.

* ENDP :-
Marks the end of produ procedure name previously begun with PROC.

Syntax :-
name ENDP

• answer · pdf

Remarks :-

See PROC

☆ END :-

directive for END of FILE command. That's END of FILE. Here as you are using "main" you have to end it with.

- Simply know that the Assembler need END directives to end the file. END can be written without Main just end the file with END.

Now ENDP denotes END of PROCEDURE here procedure id "main" so, before ending the file, End the "main" procedure. you have to end the procedure.

28) Give examples of three different instruction mnemics having zero, one, and two operands.

Ans One Operands :-

```
inc eax          ; register
inc myByte       ; memory.
```

- Two operands :-

```
add ebx, eax     ; register , register
sub myByte, 35   ; memory , constant
add eax, 36 * 35 ; register, costant-expression
```

- Zero operands :-

```
stc              ; set carry flag.
```

39) Write a program that defines symbolic names for several string literals (character b/w quates). use each symbolic name in a variables definition.

Ans :-

```
.386
.model flat, stdcall
.stack 4096
ExitProcess PROTO,
dwExit code : DWORD
Str 1 EQU <" Hadiqa", 0>
Str 2 EQU <" Iqra", 0>
Str 3 EQU <" Aqsa", 0>
.data
```

```
First   BYTE  Str 1.
Second  BYTE  Str 2
third   BYTE  Str 3

• code
  main PROC

  INVOKE ExitProcess,0
  main ENDP
  END Main.
```

30) Write a program that contain two instruction (1) add the number 5 to the EAX register and (2) add 5 to the EDX register. Generate a listing file and examine the machine code generated by the assembler. what the different, if any, did you find b/w the two instruction.

Ans 1. :-

- add the number 5 to the EAX register
- add 8 to the EDX register

* Submit the following :-

- Last name 1 . asm
- whatever the name - ~~first~~ 1st
- answer . pdf

a) EAX = -val2 +7 - val3 + val1
Assume that val1, val2 and val3 are 16-bit number integer variables.

Submit the following:-

- Lastname2.asm.

(31) Write a Program that calculate the following expression using register.

$$A = (A+B) - (C+D)$$

Assign integer value to the EAX, EBX ECX and EDX register.

Ans :-

```
.386
.model flat, stdcall
.stack 4096
ExitProcess PROTO, dwExitCode : DWORD
.
.code
main PROC

    mov eax, 3h
    mov ebx, 8h
    mov ecx, 1h
    mov edx, 8h
```

```
add   eax , ebx
add   ecx , edx
Sub   eax , ecx

INVOKE ExitProcess , 0
main   ENDP
END   main
```

38)