Assignment #03

Q1:- Using the value -35, write it as an integer literal in decimal octal and binary formats that are consistent with MASM syntax.

Ans:- -35d . 00h . 3350 . 1101110110

Q2:- Create a sigle integer expression that uses all the operators. Calculate the value of the expression.

Ans:- (5+1)(-2+3) * 2 mod 5=2

Q3:- Write the real number ~~-6.2~~ -6.2×10^4 as a real number literal using MASM syntax.

Ans:- -62E +04

Q4:- Which statemant halts the assembly language program?

Ans:- The exit statement (indirectly) call a predefined MS window function that halts the program. The ENDP direct maites the end of the main procedure. The END main direct maite the last line of the program to be assembled.

Q5:- What is a calling convention and how is used in assembly language declaration?

Ans:- A calling convention determines how parameters are passed to subroutine and how the stack is restored after the subroutine call.

Q6:- How are data label and code label different?

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

Q7:- 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.

Q8:- How ~~one~~ is a source file different from a listing file?

Ans:- A source file is given as input the assembler.
A listing file has ~~adding~~ editing text that will not assemble by the assembler and it is optionally generated.

Q10:- 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.

Q11:- Which operating system components reads and executes program?

Ans:- A monolithic kernal runs all the operating system instruction in the same address space for speed. A microkernal run most processes in user space for modularity. This central component of a computer system is

responsible for running or executing programs.

Q12:- Declare an unsigned 16-bit integer variable named wArray that uses three initializers.

Ans:- wArray word 10, 20, 30.

Q13:- Declare a string variables containing the name of your favorite color. initialize it as a null-terminate string.

Ans:- my color BYTE "Red" 0.

Q14:- 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 name ArraySize:

mArray WORD 20 DUP(?)

Ans:- Array Size = ($ - myArray)

Q15:- Show how to calculate the number of elements in the following array, and assign the value to a symbolic constant named ArraySize:

mArray DWORD 30 DUP(?)

Ans:- Array Size = ($-myArray / TYPE DWORD)

Q16:- Create an uninitialized data declaration for a 08-bit, 16-bit and 32-bit unsigned and signed integer.

Ans:-

★16-bit signed integer.

Var 1 SWORD

★8-bit unsigned integer

Var 2 BYTE

★32-bit signed

SDWORD

Q17:- Create a data definition for a doubleword that stored it in memory in little endian format.

Ans:- The least signification 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 pattern 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 : $2^{7} \ldots 2^{0}$.

Q 18:- Show the order of individual bytes in memory (lowest to highest) for the following doubleword.

variable : val1 DWORD
8765432Ih

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

var 1 BYTE 0Ah. 255

write an assembly language program to compute that following expression.

AL = BL + val1
val1 it an 8-bit variable

Q19:- Differentiate between equal-sign directive and EQU directive.

Ans:- 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 defined.
→ Name is called a symbolic constant

EQU-directive.

→ Define a symbol as either an integer or text expression.

→ Cannot be redefined.

Q20:- Name the four basis parts of an assembly language instruction.

Ans:- [label:] Mnemonic [Operands] [Comment]

Q21:- Show an example of a block comment?

Ans: COMMENT:

First line comment
Second line comment

Q22: Why it is not good idea to use numeric addresses when writing instructions that access variable?

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

Q23:- What type of argument must be posed to the Exit process procedure?

Ans: An integer, preferably 0.

Q24:- What type of argument must be passed to the Init process procedure?

Ans: make small
386
Stock 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 , 02h
mov dl , days
add dl , 30h
int 21h

```
mov ah, 4ch
int 21h

end main
```

Q25:- Declare an array of 120 uninitialized unsigned doubleword value.

Ans. Array DWORD 120(?)

Q26:- 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'

Q27:- Find out if you can declare a variable of type DWORD and assign it a negative value.

Ans: var DWORD 12345678h ; unsigned
var3 DWORD 20 DUP(?) ; Unsigned

Q28:- Discuss the following MASK directives.

Stack:-

The stack directive tell how many bytes of memory to reserve for the runtime stack

4096 happen to corresponds to the size of a memory page in this processor system for managing memory.

MODEL:-

This tells the assembler wich memory model to use in 32-bit program. We use the flat memory model which is associated with the processor's protected model.

386:-

```
-> model flat
-> Stack 4096
ExitProcess PROTO
dwExitCode : DWORD
```

The 386 directive identifies it as a 32-bit program. Line 2 uses the flat memory model and window requires the model convension to be used.

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

CODE:-

```
code
main PROC
```

it is the beginning of the code area of the program (meaning what's after-word is usually the main procedure.

Data:-

The DATA directive creates a near data segment.

→ This segment contain the frequency used data for your program

→ Data segment can occupy

* up to 64k in MS-DOS
* or upto 512 mega-byte under flat model in window NT

PROTO:-

The proto directives by using the invoke directives

Syntax:-

```
Lable PROTO [[distance]] [[Language-type]]
[[parameter]] : tag...]]
```

parameter:-

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

PROC:-

MASK starts and ends of a procedure block called label. The statement in the block can be called with the call instruction or INVOKE directives

ENDP:-

Mark the end of procedure name previously begun with PROC.

END:-

Directives for END of file command. That's end of file here as you are using "main" you have to end it with.

Q29:- Give examples of three different instruction mnemonics having Zero, one and two operands.

Ans:

one operand:

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

Two operands :-

```
add ebx, eax ; register, register
Sub myByte, 25 ; memorey, constant
add eax, 36 * 25 ; register, constant
                   expression
```

Zero Operands :-

```
stc ; set carry flag
```

Q 30 :-

Write a program that defines symbolic names for several string suterals (character between quates). Use each symbolic name in a variables defination.

Ans :

```
• 386
• model flat stdcall
• Stack 4096
  Exit process PROTO.
  dw Exit code : DWORD
  Str 1 EQU < "Hadiqa". 0>
  str 2 EQU < "Iqra". 0>
  Str 3 EQU < "Aqsa". 0>
• data
```