Saeed Ahmad Asad
14273
BS(CS)

Page No. 1

Q1:- $(64)_{10} = (?)_2$

2	64
2	32 - 0
2	16 - 0
2	8 - 0
2	4 - 0
2	2 - 0
	1 - 0

$(64)_{10} = (1000000)_2$

Q1:- (b) $(01111111)_2 = (?)_{10}$

$(0 \times 2^7) + (1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3)$
$(1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$

$= 0 + 64 + 32 + 16 + 8 + 4 + 2 + 1$
$= (127)_{10}$ Ans.

Q1:- (c) ~~4DE~~ $(4D7F)_{16} = (?)_{10}$

$= 4 \times 16 + D \times 16 + 7 \times 16 + F \times 16$
$= 4 \times 16^3 + 13 \times 16^2 + 7 \times 16^1 + 15 \times 16^0$
$= 4 \times 4096 + 13 \times 256 + 7 \times 16 + 15$
$= 16384 + 3328 + 112 + 5$
$= 19839$

Scanned with CamScanner

Saeed Ahmad Asad
14273
BS(CS) 5th Semester

Q4:- (d) $(128)_{10} = (?)_{16}$

16	128
	8 — 0

$(80)_{16}$

Q4:- (e) $(3A6F) = (?)_2$

3 A 6 F

0011 1010 0110 1111

$(0011101001101111)_2$

Q4:- (f) $(11000011110010 1)_2 = (?)_{16}$

12 3 14 5

$(C3E5)_{16}$ Ans.

Q4:- (g) $(11111111)_2 = \pm(?)_{10}$

MSB is 1 shows number is negetive so:

$-1\times2^7 + (1\times2^6) + (1\times2^5) + (1\times2^4) + (1\times2^3) + (1\times2^2) + (1\times2^1) + (1\times2^0)$

$= -1\times128 + (1\times64) + (1\times32) + (1\times16)$

Scanned with CamScanner

$+(1\times 8)+(1\times 4)+(1\times 2)+1\times 1$

$-128+127$

$=-1$ Ans.

Q1:- (h) $(-16)_{10} = (?)_2$

2	16
2	8 – 0
2	4 – 0
2	2 – 0
	1 – 0

$(00010000)_2$

Taking 2's complement

```
00010000
11101111
       +1
--------
11110000
--------
```

2's complement

$(11110000)_2$ Ans.

Q1:- (i) $(01111111)_2 - (00000111)_2$

~~1st 1's complement~~

```
01111111
00000111
--------
01111000
--------
```

Scanned with CamScanner

2's complement of (01111000)

```
    1 1 1
  10000111
      +1
  ---------
  10001000
```

$(10001000)_2$ Ans.

Q 1:- (j) $6D_{16} - 3F_{16}$

0110 1101 0011 1111

6D = 0110 1101

3F = 0011 1111

Taking 1st number of second number

01111111

1000000

Now add both numbers

```
  1101101
  1000000
 (1)0101101
```

carry

add carry in result

```
  0101101
     +1
  ---------
  0101110
```

$(2E)_{16}$ Ans.

Scanned with CamScanner

Q2:- Write short note on each of the following:

(a) Embedded systems:-
Embedded system is a combination of computer sys software and hardware which is either fixed in capability or programmable. An embedded system can be either an independent system, or it can it can be a part of a large system. It is mostly designed for a specific function or functions within a larger system.

(b) Device driver:-
Most commonly known as a driver, a device driver or a hardware driver is a group of files that enable one or more hardware devices to communicate with the computer's operating system. Without drivers, the computer would not be able to

Scanned with CamScanner

to hardware devices, such as a printer.

(c) Virtual machine concept:-

A virtual machine (VM) is a software program or operating system that not only exhibits the behaviour of a seperate computer, but is also capable of performing tasks such as running applications and programs like a separate computer. A virtual machine, usually known as a guest is created within another computing environment refferred as a "host".

(d) Instruction execution cycle:-
Instruction execution means a program to be executed by a processor consists of a set of instructions stored in memory.

Scanned with CamScanner

In instruction execution cycle the time period during which one instruction is fetched from memory and execute when computer given an instruction in machine language.

→ Each instruction is further devided into sequence of phases.

→ After the execution of program counter is incremented to point to the next instructions.

(e) Motherboard chipset :-

A chipset is a group of interdependent motherboard chips or integrated circuits that control the flow of data and instructions between the central processing unit (CPU) or microprocessor and external devices. A chipset controls external buses, memory and some peripherals. A CPU unable to function without impeccable chipset timing.

Scanned with CamScanner

(f) Access levels for input-output operations:-

1) High level operations
→ Is more flexible and usually more conveniet.
→ Hides complexity from the programming.
→ When using the I/O functions high-level I/O is slower as compared to the low-level I/O.

2) Low-level operations:
→ provides direct access to files and devices.
→ Is complex (Buffer management is to done by the programmer)
→ When using I/O functions, low-level I/O is faster as compared to the high-level I/O.
→ Uses a "file descriptor" to track the status of the file.

Scanned with CamScanner

(9) Basic parts of an assembly language instruction

An assembly instruction has 4 basic parts:

1) Label
2) Mnemonic
3) Operands
4) Comments

Labels:-

There are two types of labels:

→ Data labels
→ Code labels

Data labels:-

Data labels are just like variables in any other language. for example, C language instruction: int myVar10. Similarly, in assembly language we use data labels as variables which hold values for us. For example count is a variable in the following instruction

count DWORD 100

Data label declare in the data segment code.

Scanned with CamScanner

Code Label:-

Code labels are just the names which are used by other instructions to jump from particular position to that position where data label is located. It must end with a semicolon(;).

For example:

```
target:
  mov  ax, bx
  ...
  jmp target
```

target is a data label, which is called by the jmp instruction.

Mnemonic:-

A small word that acts as an identifier for the instructions. The mnemonics are written in code segment. In following examples mov, sub, add, jmp, call, and mul are the mnemonic:

Scanned with CamScanner

- mov => Move/assign one value to another
- sub => Subtract one value from another
- add => adds two values
- jmp => Jump to a specific location
- Call => Call a procedure/module
- mul => Multiply two values.

Operands:-

Operand is a remaining part of the instruction. There can be 0 to 3 operands. Where each operand is either a memorey operand, a register, an input/output port, or a constant expression.
Following are the operands types:

- Constant expression
- Constant
- memorey
- register

Comments:-

You should have used comments in your programs before. As you know there are two types of comments: 1. Singl-line and

Scanned with CamScanner

2. Multi-line comments. In assembly single-line comments begin with; (semicolon) whereas, multi-line comments begin with a comment directive and symbolic/character selected by the by the programmer itself and also end with the same symbol/character.

Scanned with CamScanner

Q3: (a) Differentiate Between each of following:

(a) Assembly language and High-level language:-

1) → In assembly language programs written for one processor will not run on another type of processor.
→ In high level language programs runs independently of processor type

2) Performance and accuracy of assembly language code are better than a high-level.
→ High-level languages have to give extra instructions to run code on computer.

(3) Code of assembly languages ~~have~~ is difficult to understand and debug than a high-level
→ One or two statements of high-level languages into many assembly languages codes.

Scanned with CamScanner

4) Assembly language can communicate better than a high-level language Some type of hardware actions can only be performed by assembly language.

→ In assembly language we can directly read pointers at a physical address which is not possible in high-level.

(b) Protected mode and real address mode:-

The 80286 introduced something into the x86 architecture called "protected mode".

protected mode differed from the original mode of the 8086, which was later dubbed "real mode", in that areas of memory could be physically isolated by the processor itself to prevent illegal writes to other programs could be running in memory at the same time, but any program could access

Scanned with CamScanner

any area of memory and, therefore, if malicious or errant, for example, could take down the entire system.

(c) Assembler and linker

The main output produce by assembler an input assembly language source file is the translation of the file into an produced by the assembler are relocatable files that hold code and / or data they are input file for linker.

(d) Code label and data label :-
Data label is the label that we use to define data as we defined memory locations num1 num2 ... etc in our programe.
Code label is the label that we have on code as we see in case of conditional jump (Label1) and is normally used for loop control statement.

Scanned with CamScanner

(f) Line comment and block comment

→ A single line comment and as applied as only applied sigle to a single line in the "source code"

→ A block comment usually refers to a paragraph of text. A block comment has a start symbol and an end symbol and everything between is ignored by the computer.

(g) Equal-sign directive and EQU directive:-

	Equal-sign	EQU
→	Expression is 32-bit integer	Define a symbol as either an integer or text.
→	May be redefined	Cannot be redefined
→	name is called a symbolic constant	Syntax:-
	Syntax:	name EQU expression ; integer expression
	name=expression	name EQU symbol ; existing symbol name
		name EQU <text> ; any text

Scanned with CamScanner

Q4: (a) Explan the concept of portability as it applies to programming languages.

Ans: A language whose source program can be compiled and run on a wide variety of computer system is said to be portable.

Q4: (b) Why would a high-level language not be an ideal tool for writing a program that directly accesses a particular brand of printer?

Ans: A high-level language may not be provide for direct hardware access. Even it does awkward coding technique must often be used, resulting in possible maintain problem.

Q4: (c) Why was unicode invented?

Ans: Unicode is a universal computing standard to represent texts in most writing system. It was invented to store most of the

Scanned with CamScanner

words character. it is stated during 1987.

Q4:- (d) If W = 11101100, X = 00010011, and Y = 00111100, then find Z = W ∨ X ∧ ¬Y.

Ans:

Z = W ∨ X ∧ ¬Y

Y = 00111100

¬Y = 11000011

⇒ X ∧ ¬Y = 00010011
11000011
AND 00000011

W ∨ X ∧ ¬Y

11101100
00000011
OR 11101111

Q4:- (e) Create a truth table to show all possible inputs and outputs for the Boolean function described by ¬(A ∨ B)

¬(A ∨ B)

A	B	A ∨ B	¬(A ∨ B)
F	F	F	T
F	T	T	F
T	F	T	F
T	T	T	F

Q4:- (f) Why does memory access take more machine cycles than register access?

Ans: Conventional memory is outside the CPU, and it responds more slowly to access requests. Registers are hard-wired inside the CPU.

Scanned with CamScanner

Q4:- (g) Discuss the basic program execution registers used in x86 32-bit processors.

Ans:-

XMM Registers:-
The x86 architecture also contain eight 128-bit register called streaming SIMD extension to the instruction.

MMX Registers:-
MMX technology improves the performance of intel processor when implementing advanced multimedia and communication application. The 64-bit MMX register support special instruction called SIMD (single-instruction) Multiple data.

Segment Register:-
In real address mode, 16-bit segment register indicate base addresses of preassigned memory areas named segments.

Scanned with CamScanner

Basic programming Execution Register :-

Registers are high-speed storage location inside the CPU designed to be accessed at much higher speed than conventional memory when a processing loop is optimized for speed.

Scanned with CamScanner

Q28:- Discuss the following MASK directives.

Stack:-

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

Scanned with CamScanner

4096 happen to correspond 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
Exit process PROTO
dw Exit code : 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.

Scanned with CamScanner

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...]
```

Scanned with CamScanner

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 statements 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.

Scanned with CamScanner

Q6:(a) Write a program that calculates the following expression: A=(A+B)-(C+D)

Ans: A = (A + B) - (C + D)

```
data ; data segment, read and write
Var A  BYTE 10
Var B  BYTE 20
Var C  BYTE 30
Var D  BYTE 40

 final var BYTE

code ; code segment, read-only
     main PROC
MOV al , Var A
Mov bl , Var B
Mov cl , Var C
Mov dl , Var D

ADD al , bl ; compute (A+B)
ADD cl , dl ; Compute (A+B)
SUB al, cl ; Compute (A+B)-(C+D)
   Mov final val, al
call Dump Regs
```

Scanned with CamScanner

```
exit
main ENDP
END main
```

Q6:- (b) Show the order of individual bytes in memory for the following doubleword variable using little endain order:

Ans: When placed in memory at offset 0000, 78h would be stored in the first byte 56h would be stored in the second byte and the remain bytes would be at offset 0002 and 0003 as show in figure:

12345678h

0000	78
0001	56
0002	34
0003	12

Scanned with CamScanner

Q6:- (C)

```
.data
string byte "Assembly language is easy", 0
String size byte ?

.code
mov eax size OF string 1
mov stringsize , eax
```

Q6: (D) Let the Arthematic operation is $x = -a + (b - c)$

```
INCLUDE vine 32.inc

• data
X DWORD ? ; uninitialized variable
a DWORD 10 ; initialize variable 'a'
b DWORD 20 ; initialize variable 'b'
C DWORD 15 ; initialize variable 'c'

• Code
main PROC
mov eax, a ; move value of 'a' int 'eax'
neg eax ; EAX = -10
mov ebx, b ; mov value of b into 'ebx'
Sub ebx, C ; EBX = 11
```

Scanned with CamScanner

page No. 29

```
add eax, ebx ; perform -10 + 11
mov x, eax ; 1

call DUMPRegs
exit
main ENDP
END main
```

Scanned with CamScanner