Id No 13943

Name Mehran Ali Shah

Subject MAL

Question No 1:

Solve each of the following

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

b. 01111111_2 = (127)$_{10}$

c. $4D7F_{16}$ = (19839)$_{10}$

d. 128_{10} = (80)$_{16}$

e. $3A6F_{16}$ = (11101001101111)$_2$

f. 1100001111100101_2 = (C3E5)$_{16}$

g. 111111112 = ± (255)10 hint: [use 2's complement form]

h. -1610 = (-10000)2 hint: [use 2's complement form]

i. $01111111_2 – 00000111_{2\ =\ 01111000}$ hint: [use 2's complement form]

j. $6D_{16} – 3F_{16\ =\ 0101110}$ hint: [use 2's complement form]

Question No 2:

Embedded systems

Answer:

An embedded system is a combination of computer hardware and software designed for a specific function or functions within a larger system.

The systems can be programmable or with fixed functionality.

Device driver

Answer:

More commonly known as a driver, a device driver or 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 send and receive data correctly to hardware devices, such as a printer.

Virtual machine concept

Answer:

A virtual machine (VM) is an image file managed by the hypervisor that exhibits the behavior of a separate computer, capable of performing tasks such as running applications and programs like a separate computer.

OR

In other words, a VM is a software application that performs most functions of a physical computer, actually behaving as a separate computer system.

Instruction execution cycle

Answer:

A single machine instruction does not just magically execute all at once. The CPU has to go through a predefined sequence of steps to execute a machine instruction, called the instruction execution cycle. Let's assume that the instruction pointer register holds the address of the instruction we want to execute.

Here are the steps to execute it:

One cycle

1. First, the CPU has to fetch the instruction from an area of memory called the instruction queue. Right after doing this, it increments the instruction pointer.

2. Next, the CPU decodes the instruction by looking at its binary bit pattern. This bit pattern might reveal that the instruction has operands (input values).

3. If operands are involved, the CPU fetches the operands from registers and memory. Sometimes, this involves address calculations.

4. Next, the CPU executes the instruction, using any operand values it fetched during the earlier step. It also updates a few status flags, such as Zero, Carry, and Overflow.

5. Finally, if an output operand was part of the instruction, the CPU stores the result of its execution in the operand.

Motherboard Chipset

Answer:

A motherboard chipset is a collection of processor chips designed to work together on a specific type of motherboard.

Various chipsets have features that increase processing power, multimedia capabilities, or reduce power consumption.

Access levels for input–output operations

Answer:

Application programs routinely read input from keyboard and disk files and write output to the screen and to files.

I/O need not be accomplished by directly accessing hardware—instead, you can call functions provided by the operating system.

I/O is available at different access levels, similar to the virtual machine concept shown in Chapter 1. There are three primary levels:

• High-level language functions: A high-level programming language such as C++ or Java contains functions to perform input–output.

These functions are portable because they work on a variety of different computer systems and are not dependent on any one operating system.

• Operating system: Programmers can call operating system functions from a library known as the API (application programming interface).

The operating system provides high-level operations such as writing strings to files, reading strings from the keyboard, and allocating blocks of memory.

• BIOS: The basic input–output system is a collection of low-level subroutines that communicate directly with hardware devices.

The BIOS is installed by the computer's manufacturer and is tailored to fit the computer's hardware. Operating systems typically communicate with the BIOS.

Basic parts of an assembly language instruction

Answer:

Instruction:

Instructions are statements that execute when we assemble the program. The instructions (written in assembly language) are translated into machine language bytes.

An assembly instruction has 4 basic parts:

1. Label	(optional)
2. Mnemonic	(required)
3. Operand	(depends on the instruction)
4. Comment	(optional)

Question No 3:

Assembly language and high-level language

Answer:

HLL (High Level Language) programs are machine independent. They are easy to learn, easy to use, and convenient for managing complex tasks.

Assembly language programs are machine specific. It is the language that the processor directly understands.

High Level Language:

It is programmer friendly language.

High level language is less memory efficient.

It is easy to understand.

It is simple to debug.

It is simple to maintain.

It can run on any platform.

It needs compiler or interpreter for translation.

Assembly language (Low Level Language) :

It is a machine friendly language.

Low level language is high memory efficient.

It is tough to understand.

It is complex to debug comparatively.

It is complex to maintain comparatively.

It is machine-dependent.

It needs assembler for translation.

Protected mode and real address mode

Answer:

A 'real mode' program uses BIOS subroutines along with OS subroutines whereas a 'protected mode' program uses only OS subroutines.

Assembler and linker

Answer:

Assembler:

An assembler then translates the assembly program into machine code (object).

Linker:

A linker tool is used to link all the parts of the program together for execution (executable machine code).

Instruction and directive

Answer:

A directive is mainly an order, usually issued by an authority. A directive may establish policy, assign responsibilities, define objectives and delegate authority to those working in and with the authoritative figure. Instructions, on the other hand, act as guidelines.

Code label and data label

Answer:

Data Label is the label that we use to define data as we defined memory locations num1,num 2 ....etc. in our programs. Code Label is the label that we have on code and is normally used for loop control statements.

Line comment and block comment

Answer:

The first is called a *single line comment* and, as implied, only applies to a single line in the "source code" (the program). The second is called a *Block* comment and refers 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.

Question No 4:

Explain the concept of portability as it applies to programming languages

Answer:

Portability, in relation to software, is a measure of how easily an application can be transferred from one computer environment to another. A computer software application is considered portable to a new environment if the effort required to adapt it to the new environment is within reasonable limits. The meaning of the abstract term 'reasonable' depends upon the nature of the application and is often difficult to express in quantifiable units. Portability is a form of reusability. Some kinds of software are known to be less portable than others. An example of software that is not portable would be assembly code, since assembly code is specific to processor type. No software is perfectly portable because all software have limitations. Some programming languages are fairly portable, for example the C language. C compilers are readily available for the majority of operating systems, which in turn makes C programs very portable. This portability of C language programs has resulted in some programmers re-writing their programs and recompiling them in C to make them much more portable. Portability is also used to describe the flexibility of the use of data. Some file formats are less portable than others. For example, to view files with file formats such as PDF or JPEG, the formats depend on the availability of appropriate software applications.

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

Answer:

A high-level language many not provide for direct hardware access. Even if it does, awkward coding techniques must often be used, resulting in possible maintenance problems.

Why was Unicode invented?

Answer:

At earlier times, being an electronic device, a computer stores data converting into corresponding numbers. There were several character encoding systems for mapping characters to some ASCII values or numbers. For different languages, numerous encoding is required for the letters, punctuation, and other symbols. Hence, two characters could use the same number or even two numbers were used for a single character. Hence, before discovering the Unicode scheme, there may be a conflict within these traditional encoding schemes. Unicode was discovered to resolve this issue. Unicode is an established worldwide standard for data representation that can be used for numbers, characters, mathematical symbols, and all other characters. A unique number (8/16/32-bit) between U+0000 to U+10FFFF is available for encoding text or numbers from all the languages. For example, U+0042 represents the English letter "B."

Discuss the basic program execution registers used in x86 32-Bit processors.

Registers: high-speed memories located in the CPU

Registers for 8086 and are 16 bits wide

Registers for IA-32 family are 32 bits wide

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

Why does memory access take more machine cycles than register access?

Answer:

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

Question No 6:

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

```
.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
```

Show the order of individual bytes in memory for the following doubleword variable using little endian order:

dval DWORD 12345678h

Write a program that performs arithmetic operations on different register operands and stores the result in memory. Give stepwise explanation of each statement.

```
N=12 //loop index initial value
ADDC(r31,N,r1)  //r1=loop index
ADDC(r31,1,r0) //r0=accumulated product
loop:MUL(r0,r1,r0)  //r0=r0*r1
SUBC(r1,1,r1)  // r1=r1-1
BNE(r1,loop,r31) //if r1!=0, NextPc=loop
```