**Program: BC (CS)**

**Subject: Microprocessor & Assembly Language**

**Major Assignment Mid-Term**

**Course Code: CSC-304**

**EDP Code: 102007054 Semester: Summer 2020**

**Q.1** Solve each of the following:

1. 6410 = ( 1000000)2
2. 011111112 = ( 127)10
3. 4D7F16 = ( 19839 )10
4. 12810 = ( 80 )16
5. 3A6F16 = ( 11101001101111 )2
6. 11000011111001012 = ( C3E5 )16
7. 111111112 = ± ( 255 )10 hint: [use 2’s complement form]
8. -1610 = ( -10000 )2 hint: [use 2’s complement form]
9. 011111112 – 000001112 = 01111000 hint: [use 2’s complement form]
10. 6D16 – 3F16 = 0101110 hint: [use 2’s complement form]

**Q.2** Write short note on each of the following:

1. Embedded systems

Definition

An embedded system is a microprocessor- or microcontroller-based system of hardware and software designed to perform dedicated functions within a larger mechanical or electrical system.

An embedded system is a microprocessor-based computer hardware system with software that is designed to perform a dedicated function, either as an independent system or as a part of a large system. At the core is an integrated circuit designed to carry out computation for real-time operations

1. Device driver

A device driver is a code which manages and tunes the hardware parameters in a card. This is done in the deeper level of the hardware system. This code is written by specialists that know exactly how the card functions, how the computer interrupt system functions and all of that material (electronics and software). Essentially it is written in C and possibly in assembly. As an example, we can mention the one which handles the keyboard or the mouse. For each peripheral we need a driver.

1. Virtual machine concept

A VirtualMachine is a program that translates a standard series of instruction codes into hardware specific MachineLanguage. In recent years the term has come to have the connotation of a software platform that exists on multiple hardware platforms and provides an abstraction layer that allows programmers to write code without considering the hardware that it will run on.

A virtual machine is a hypothetical device that is capable of executing a specific instruction set. Such an instruction set is commonly called ByteCode. A common implementation strategy for programming languages is to compile code into instructions for a specialized VirtualMachine. Subsequently, these instructions can be actually executed in a number of different ways:

Interpretation by software

Translation into the instruction set of a real processor

Direct interpretation in hardware

1. Instruction execution cycle

Instruction Execution means a program to be executed by a processor consist of set of instructions stored in memory.

 **Instruction Execution Cycle**

The time period during which one instruction is fetched from memory and execute when computer gives an instruction in machine language.

Each Instruction is further divided into sequence of phases.

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

1. 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 cache and some peripherals. A CPU is unable to function without impeccable chipset timing.

A chipset includes the circuit board layout/functionality and circuit mechanisms. Varieties include microprocessors and modem card chipsets. In addition, a CPU has several different chipsets that vary according to architecture.

A chipset is specifically designed for a motherboard. The chipset and motherboard must be compatible with the CPU to prevent system failover. Most chipset drivers are manually updated and installed.

A chipset has two sections – southbridge and northbridge – with specific sets of functions that communicate between the CPU and external devices

1. Access levels for input–output operations
2. Basic parts of an assembly language instruction

It has four parts; label, mnemonic, operand, comment; not all are present in every line.

The first part (LOOP in this example) is a label ; this is a word, invented by the programmer, which identifies this point in the program.

The second part is the mnemonic .This corresponds to a particular kind of instruction

The third part of the line is an operand

The last part of the line is a comment. This does not affect the actual instruction at all; it is not part of the instruction, and is not assembled; instead it helps the programmer to remember what this part of the program does.

1 of 2

Q.**3** Differentiate between each of the following:

1. Assembly language and high-level language

 HIGH LEVEL

Formal structures make it easy to organize and maintain large sections of code. The language may not provide for direct hardware access. Even if it does, awkward coding techniques may be required,resulting in maintenance difficulties Usually portable. The source code can be recompiled on each target operating system with minimal changes. May produce large executable files that exceed the memory capacity of the device.

ASSEMBLY LANGUAGE

Minimal formal structure, so one must be imposed by programmers who have varying levels of experience. This leads to difficulties maintaining existing code.

Hardware access is straightforward and simple. Easy to maintain when programs are short and well documented

Must be recoded separately for each platform, using an assembler with a different syntax. Difficult to maintain.

1. Protected mode and real address mode

The main difference is the mode the cpu is in. In protected mode the OS can use features like paging and virtual memory. Also real mode code is never in 32 bits whereas protected mode code can be 16 bits or 32 bits. ... The protected mode, on the other hand, has virtual memory feature enabled.

1. Assembler and linker

**Assembler:** A computer will not understand any program written in a language, other than its machine language. The programs written in other languages must be translated into the machine language. Such translation is performed with the help of software. A program which translates an assembly language program into a machine language program is called an assembler. If an assembler which runs on a computer and produces the machine codes for the same computer then it is called self assembler or resident assembler. If an assembler that runs on a computer and produces the machine codes for other computer then it is called Cross Assembler.

Assemblers are further divided into two types: One Pass Assembler and Two Pass Assembler.

**Linker:** In high level languages, some built in header files or libraries are stored. These libraries are predefined and these contain basic functions which are essential for executing the program. These functions are linked to the libraries by a program called Linker. If linker does not find a library of a function then it informs to compiler and then compiler generates an error. The compiler automatically invokes the linker as the last step in compiling a program.

Not built in libraries, it also links the user defined functions to the user defined libraries. Usually a longer program is divided into smaller subprograms called modules. And these modules must be combined to execute the program. The process of combining the modules is done by the linker.

1. Instruction and directive

The main difference between directions or instruction is the fact that 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. A directive may establish or describe a policy, a program and/or an organization. For example: directive to set up a government entity, directive to set up a local subsidy, directive to set up an overseas office, directive to reduce wastage, directive for recycling, etc.

Instructions, on the other hand, act as guidelines. They often appear as a series of steps or stages one must complete one after the other. Instructions are mainly associated with instruction or teaching. Hence, instructions are given to teach somebody something. For example: instructions for cooking, instructions for knitting a scarf, instructions for building a dollhouse, instructions for completing a project, instructions for writing a report, instruction manual for any and all electronic appliances, etc.

1. 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 programs. Code Label is the label that we have on code as we see in case of conditional jump (Label l1) and is normally used for loop control statements.

1. Line comment and block comment

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.

1. Equal-sign directive and EQU directive

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

The. syntax is. name = expression.

The EQU directive gives a symbolic name to a numeric constant, a register-relative value or a PC-relative value.

**Q.4** Give answer to each of the following

1. Explain the concept of portability as it applies to programming languages.

Portability is a characteristic attributed to a computer program if it can be used in an operating systems other than the one in which it was created without requiring major rework. Porting is the task of doing any work necessary to make the computer program run in the new environment.

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

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.

1. Why was Unicode invented?

Unicode, international character-encoding system designed to support the electronic interchange, processing, and display of the written texts of the diverse languages of the modern and classical world.

1. If W = 11101100, X = 00010011, and Y = 00111100, then find Z = W ∨ X ∧ ￢Y.
2. Create a truth table to show all possible inputs and outputs for the Boolean function described by￢( A ∨ Β)
3. Why does memory access take more machine cycles than register access?

Because of different in speeds of cpu the system bus and the memory circuit.

1. 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.

* 1. Discuss the following MASM directives in detail:

 INCLUDE .386 .MODEL .STACK PROTO

 .DATA .CODE PROC ENDP END

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

main PROC

MOV eax, varA

MOV ebx, varB

MOV ecx, varC

MOV edx, varD

SUB eax, ebx; Compute(A - B)

SUB ecx, edx; Compute(C - D)

ADD eax, ecx;Compute (A-B)+(C-D)

MOV varA,eax

1. Show the order of individual bytes in memory for the following doubleword variable using little endian order:  *dval DWORD 12345678h*
2. Write a statement that causes the assembler to calculate the number of bytes in the following string, and assign the value to a symbolic constant named StringSize: *string1 byte “Assembly language is easy”, 0*
3. 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

2 of 2