***ASSIGNMENT#01***

**Name: Muhammad Tahir**

**Roll No: 21k-4503**

**Sec: 3E-BSCS**

**Question#01**

**1. Explain the contents of Segment registers in Real memory addressing mode and Protected mode.**

The code segment holds the base address to pre assigned memory in the program. In real mode operation, it defines the start of a 64K-byte section of memory; in protected mode, it selects a descriptor that describes the starting address and length of a section of memory holding code.

**2. Why does memory access take more machine cycles than register access?**

Since memory is far from CPU, it reads more clock cycle to transmit data and these clock cycles are called wait states.

**3. Write down the name of two types of applications that would be better suited to assembly language than a high level language.**

Hardware device drivers and embedded systems and computer games requiring direct hardware access. A high-level language may not provide for direct hardware access. Even if it does, awkward coding techniques must often be used, resulting in possible maintenance problems.

**4. Describe the execution cycle of an instruction of a computer program, mention the registers and counters involved along with their functions.**

**i.** First, the CPU has to fetch the instruction from an area of memory called the instruction queue. It then increments the instruction pointer.

**ii.** 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).

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

**iv.** 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.

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

**vi.** An operand is a value that is either an input or an output to an operation.

**vii.** In order to read program instructions from memory, an address is placed on the address bus.

**viii.** Next, the memory controller places the requested code on the data bus, making the code available inside the code cache.

**ix.** The instruction pointer’s value determines which instruction will be executed next.

**x.** The instruction is analyzed by the instruction decoder, causing the appropriate digital signals to be sent to the control unit, which coordinates the ALU and floating-point unit.

**xi.** Control bus carries signals that use the system clock to coordinate the

transfer of data between different CPU components.

**5. What is the difference between a machine cycle and an instruction cycle?**

In a machine cycle, a processor repeats a set of four basic operations; fetching, decoding, executing, and sometimes storing. Fetching is the process of obtaining a program instruction or data item from memory. Decoding refers to the process of translating the instruction into signals the computer can execute. Executing is the process of carrying out the commands. Storing means writing the result to memory.

Instruction cycle is a cycle in which one instruction that is fetched from the memory and get executed right after when machine language get any instruction from the Computer.

**6. A major feature of the Java language is that compiled programs run on nearly any computer system. Why?**

The most significant feature of java is that it is platform independent meaning it can be compiled and executed on any device in the world because it uses the concept of byte code. Java compiler does not compile the source code to machine code like high level languages are compiled. Instead, it converts the source code into an intermediate code called the byte code and this byte code is further translated to machine-dependent form by another layer of software called Java Virtual Machine. Therefore, JVM can execute bytecode on any platform or operating system on which it is present, regardless of the fact that on which machine the bytecode was generated.

**7. Elaborate how the following task is achieved by a computer (Limit your answer to memory, registers and buses). ADD [12FCBD10h], AL**

Here first this instruction loads into register and then instruction pointer points towards the instruction. Then this instruction is accessed by the CPU through an instruction pointer. Then the CPU places the address on the address bus and reads the data from that address in read mode, then it accesses the data from the register and ALU calculates the addition. At the end, the CPU stores the result in write mode on the address provided through the data bus.

**8. Which of the following instructions are illegal (if any)? (Circle its letter)**

**(A)** MOV 2020h, AL

*Reason:*

Not Possible because constant cannot be a destination operand

*Correct:* MOV AL,2020h

**(B)** MOVZX AX, BX

**(C)** *Reason:*

Not Possible because registers are of the same size.

Correct: movzx eax,bx

**(D)** MOV AL, WORD PTR [EBX]

*Reason:*

Not Possible because operands are not of same type and data type should match bits with register.

Correct: MOV EBX WORD PTR [EBX] or MOV EAX WORD PTR [EBX] or MOV EAX WORD PTR [EAX] or MOV EBX WORD PTR [EAX]

**(E) ADD [AL], [CH]**

*Reason:*

Not Possible, brackets should not be placed. We cannot type cast registers.

**Correct:** ADD AL, CH

**(F) INC 1Ah**

*Reason:*

Not Possible because constant does not have a fixed memory. We have to save the value in the register first.

Correct: mov eax,1Ah

inc eax

**9. How many bytes are contained in the following declaration?**

Var WORD "AB", ABh, 20 DUP(10 DUP("AB"), 10 DUP(ABh), "AB", ABh)

WORD "AB", ABh = 2bytes + 2bytes = 4bytes

20 DUP(10 DUP("AB"), 10 DUP(ABh), "AB", ABh) = 20 x (10+10+1+1) = 440 bytes

WORD 20 DUP(10 DUP("AB"), 10 DUP(ABh), "AB", ABh) = 440bytes x 2bytes = 440bytes

Var WORD "AB", ABh, 20 DUP(10 DUP("AB"), 10 DUP(ABh), "AB", ABh) = 4 bytes + 880bytes = 884bytes

**10. Give the contents of the status flags C, O, S and Z and the content of destination register after the execution of each of the following sequence of instructions:**

**(A)** MOV AX, 8F7AH

ADD AX, 7AF8H

Carry Flag = 1

Overflow Flag =0

Sign Flag = 0

Zero Flag = 0

**(B)** MOV BX, 0FA77H

INC BX

Carry Flag = 0

Overflow Flag =0

Sign Flag = 1

Zero Flag = 0

***Question#02***

**Find the missing value (directed by “?”)**

**I. Segment: 560E h**

**Offset: 53D9 h**

**Real Address: ?**

*Answer:*

Real Address = Offset + (Segment x 10 h)

Real Address = 53D9 h + (560E h x 10 h)

Real Address = 53D9 h + 560E0 h

Real Address = 5B4B9 h

**II. Segment: 0893 h**

**Offset: ?**

**Real Address: BC893 h**

*Answer:*

Real Address = Offset + (Segment x 10 h)

BC893 h = Offset + (0893 h x 10 h)

BC893 h = Offset + 08930 h

BC893 h – 08930 h = Offset

Offset = B3F63 h

**III. Segment: ?**

**Offset: 50AD h**

**Real Address: ED32D h**

*Answer:*

Real Address = Offset + (Segment x 10 h)

ED32D = 50AD + (Segment x 10 h)

(Segment x 10 h) = ED32D - 50AD

(Segment x 10 h) = E8280

Segment = E828 h

**Question#03**

**Write assembly language code for the following:**

**i.Write a code snippet to exchange the values of two variables defined as ‘A’ and ‘B’.**

TITLE Exchange Of Values (Text.asm)

INCLUDE Irvine32.inc

.data

valueA BYTE 82h

valueB BYTE 33h

.code

main PROC

mov al,valueA

xchg al,valueB

mov valueA,al

call DumpRegs

exit

main ENDP

END main

**II. Write assembly language code that directly exchanges respective elements of two word sized arrays X1 and X2 having 20 elements each. Your code should not use a third array.**

INCLUDE Irvine32.inc

.data

X1 WORD 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20

X2 WORD 20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1

.code

main PROC

mov esi,OFFSET[X1]

mov edi,OFFSET[X2]

mov ecx,LENGTHOF X1 ;loop will execute 20 times

l1:

mov eax,[esi]

mov ebx,[edi]

add esi,2

add edi,2

mov eax,ebx

Loop L1

mov ecx,LENGTHOF X2

mov esi,0

mov edi,0

mov esi,OFFSET X1

mov edi,OFFSET X2

mov eax,0

mov ebx,0

l2:

mov eax,[esi]

mov ebx,[edi]

add esi,2

add edi,2

mov ebx,eax

Loop l2

call DumpRegs

exit

main ENDP

END main

**III. Write an assembly language program that sums an array of 100 integers of type BYTE. You may assume that the array is defined and initialized. Use LOOP instruction to design the loop.**

TITLE Exchange Of Values (Text.asm)

INCLUDE Irvine32.inc

.data

;assume array A of byte size having 100 elements

.code

main PROC

mov al,offset A

mov bl,0

mov cl,100

l1:

add bl,[al]

add cl,1

loop l1

call writeint

exit

main ENDP

END main