Program: BC (CS)

Subject: Microprocessor & Assembly Language

Assignment Number: 06

Course Code: CSC-304

EDP Code: 101902031

Spring Semester 2019

Q.1 Write a single instruction using 16-bit operands that clears the high 8 bits of AX and does not change the low 8 bits.

Q.2 Write a single instruction using 16-bit operands that sets the high 8 bits of AX and does not change the low 8 bits.

Q.3 Write a single instruction (other than NOT) that reverses all the bits in EAX.

Q.4 Write instructions that set the ZF if the 32-bit value in EAX is even and clear the ZF if EAX is odd.

Q.5 Write a single instruction that converts an uppercase character in AL to lowercase but does not modify AL if it already contains a lowercase letter.

Q.6 Which jump instructions follow unsigned integer comparisons?

Q.7 Which jump instructions follow signed integer comparisons?

Q.8 Will the following code jump to the label named **Target**?

```
mov ax,8109h
cmp ax,26h
jg Target
```

Q.9 Implement the following pseudocode in assembly language:

a.
```
if ebx > ecx
X = 1
```

b.
```
if edx <= ecx
X = 1
else
X = 2
```

Q.10 What will be the value of BX after the following instructions execute?

```
mov bx,0FFFFh
and bx,6Bh
```

Q.11 What will be the value of BX after the following instructions execute?

```
mov bx,91BAh
and bx,92h
```

Q.12 What will be the value of BX after the following instructions execute?

```
mov bx,0649Bh
or bx,3Ah
```

Q.13 What will be the value of BX after the following instructions execute?

```
mov bx,029D6h
xor bx,8181h
```

Q.14 What will be the value of EBX after the following instructions execute?

```
mov ebx,0AFAF649Bh
or ebx,3A219604h
```

Q.15 What will be the value of RBX after the following instructions execute?

```
mov rbx,0AFAF649Bh
xor rbx,0FFFFFFFFh
```

Q.16 In the following instruction sequence, show the resulting value of AL where indicated, in binary:

```
mov al,01101111b
and al,00101101b      ; a.
mov al,6Dh
and al,4Ah            ; b.
mov al,00001111b
or al,61h             ; c.
mov al,94h
xor al,37h            ; d.
```

Q.17 In the following instruction sequence, show the resulting value of AL where indicated, in hexadecimal:

```
mov al,7Ah
not al                ; a.
mov al,3Dh
and al,74h            ; b.
mov al,9Bh
or al,35h             ; c.
mov al,72h
xor al,0DCh           ; d.
```

Q.18 In the following instruction sequence, show the values of the Carry, Zero, and Sign flags where indicated:

```
mov al,00001111b
test al,00000010b ; a. CF= ZF= SF=
mov al,00000110b
cmp al,00000101b ; b. CF= ZF= SF=
mov al,00000101b
cmp al,00000111b ; c. CF= ZF= SF=
```

Q.19 Which conditional jump instruction executes a branch based on the contents of ECX?

Q.20 How are JA and JNBE affected by the Zero and Carry flags?

Q.21 What will be the final value in EDX after this code executes?

```
mov edx,1
mov eax,7FFFh
cmp eax,8000h
jl L1
mov edx,0
L1:
```

Q.22 What will be the final value in EDX after this code executes?

```
mov edx,1
mov eax,7FFFh
cmp eax,8000h
jb L1
mov edx,0
L1:
```

Q.23 What will be the final value in EDX after this code executes?

```
mov edx,1
mov eax,7FFFh
cmp eax,0FFFF8000h
jl L2
mov edx,0
L2:
```

Q.24 Will the following code jump to the label named **Target**?

```
mov eax,-30
cmp eax,-50
jg Target
```

Q.25 Will the following code jump to the label named **Target**?

```
mov eax,-42
cmp eax,26
ja Target
```

Q.26 Write a single instruction that converts an ASCII digit in AL to its corresponding binary value. If AL already contains a binary value (00h to 09h), leave it unchanged.

Q.27 Write instructions that calculate the parity of a 32-bit memory operand.

Q.28 Write instructions that jump to label L1 when the unsigned integer in DX is less than or equal to the integer in CX.

Q.29 Write instructions that jump to label L2 when the signed integer in AX is greater than the integer in CX.

Q.30 Write instructions that first clear bits 0 and 1 in AL. Then, if the destination operand is equal to zero, the code should jump to label L3. Otherwise, it should jump to label L4.

Q.31 Implement the following pseudocode in assembly language. Use short-circuit evaluation:

a.
```
if( val1 > ecx ) AND ( ecx > edx )
X = 1
else
X = 2;
```

b.
```
if( ebx > ecx ) OR ( ebx > val1 )
X = 1
else
X = 2
```

c.
```
if( ebx > ecx AND ebx > edx) OR ( edx > eax )
X = 1
else
X = 2
```

Q.32 Implement the following pseudocode in assembly language. Use short-circuit evaluation and assume that A, B, and N are 32-bit signed integers.

```
while N > 0
if N != 3 AND (N < A OR N > B)
N = N - 2
else
N = N - 1
end whle
```

Q.33 Create a procedure that fills an array of doublewords with *N* random integers, making sure the values fall within the range j...k, inclusive. When calling the procedure, pass a pointer to the array that will hold the data, pass *N*, and pass the values of j and k. Preserve all register values between calls to the procedure. Write a test program that calls the procedure twice, using different values for j and k. Verify your results using a debugger.

Q.34 Create a procedure that returns the sum of all array elements falling within the range *j...k* (inclusive). Write a test program that calls the procedure twice, passing a pointer to a signed doubleword array, the size of the array, and the values of *j* and *k*. Return the sum in the EAX register, and preserve all other register values between calls to the procedure.

Q.35 Data transmission systems and file subsystems often use a form of error detection that relies on calculating the parity (even or odd) of blocks of data. Your task is to create a procedure that returns True in the EAX register if the bytes in an array contain even parity, or False if the parity is odd. In other words, if you count all the bits in the entire array, their count will be even or odd. Preserve all other register values between calls to the procedure. Write a test program that calls your procedure twice, each time passing it a pointer to an array and the length of the array. The procedure's return value in EAX should be 1 (True) or 0 (False). For test data, create two arrays containing at least 10 bytes, one having even parity, and another having odd parity.