**FAST NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES**

KARACHI CAMPUS



Assignment 1 Fall 2021

*Computer Organization & Assembly Language*

Total Points: **144**

Solve on this paper, and attached the program results

Roll No: Section : Signature: \_\_\_\_\_\_\_\_\_\_

Part I

Question No. 1: Programming Basics [10\*2 = 20 Points]

(i) In the following instruction sequence, show the resulting value of AL where indicated, in hexadecimal:

MOV AL,7AH

NOT AL ; a. AL =

MOV AL,3DH

AND AL, 74H ; b. AL =

MOV AL,9BH

OR AL,35H ; c. AL =

MOV AL,72H

XOR AL,0DCH ; d. AL =

(ii) Differentiate between the following Assembly Language instructions:

MOV EAX, OFFSET VAR1

MOV EAX, VAR1

(iii) List *four* important uses of the runtime stacks in programs.

(iv) Suppose EAX=1234H, EBX=5678H, ECX=9ABCH, and ESP=100H, Give the contents of EAX, EBX, ECX, and ESP after the execution of the following instructions:

PUSH EAX

PUSH EBX

XCHG EAX, ECX

POP ECX

PUSH EAX

POP EBX

1. EAX : b) EBX : ECX : c) ESP :

(v) What additional instructions are generated by the assembler as a result of assembling the following procedure?

MYSUM PROC USES ESI ECX

MOV ECX, 10

L1:

ADD EAX, [ESI]

SUB ESI, 4

LOOP L1

ret

MYSUM ENDP

(vi) Which of the following instructions are illegal (if any)? (Circle its letter)

(A) PUSH AL

(B) MOVZX AX, BX

(C) MOV AL, WORD PTR [EBX]

(D) ADD [AL], [CH]

(E) INC 1Ah

(vii) How many bytes are contained in the following declaration?

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

(viii) Generate a Map file for an assembly language program that has a code size of 100h bytes, data size of 50h bytes and a stack of 200h bytes. Using this map file, give the contents of CS, DS, and SS registers if this program is loaded at address of 508A0h.

(ix) The shown program sets AH to a value depending on the comparison result of unsigned integers V1 and V2. For each condition in the table below, use “√” sign to indicate which value AH will have after the program is executed. If there are more than one possibility, use “?” sign to indicate which value of AH is possible.

.DATA

|  |  |  |  |
| --- | --- | --- | --- |
|  | AH =1 | AH=2 | AH=3 |
| If V1=V2 then |  |  |  |
| If V1<V2 then |  |  |  |
| If V1>V2 then |  |  |  |

V1 DB(?)

V2 DB(?)

.CODE

Start:

•

•

MOV AL, V1

CMP AL, V2

JZ Label1

JS Label1

MOV AH, 1

JMP Continue

Label1:

JE Label2

MOV AH, 2

JMP Continue

Label2:

MOV AH, 3

Continue:

•

•

•

1. Give the contents of the following registers, along with the run-time stack, when the following instructions are executed. Initially, consider ESP = 00001FF8h.

Note: SOLVE THIS PART HERE. No Marks will be awarded without proper working using the stack diagrams.

X1 DWORD 25H

X2 DWORD 27H

MAIN PROC

PUSH 6H

PUSH 5H

CALL P1

11500000H MOV RESULT, EAX ; ESP: \_\_\_\_\_\_\_\_\_\_\_\_

MAIN ENDP

P1 PROC

115000A4H PUSH EBP

MOV EBP, ESP ; EBP: \_\_\_\_\_\_\_\_\_\_\_\_

MOV EAX, [EBP+8]

ADD EAX, [EBP+12] ; EAX: \_\_\_\_\_\_\_\_\_\_\_\_

PUSH OFFSET X1

PUSH OFFSET X2 ; ESP: \_\_\_\_\_\_\_\_\_\_\_\_

POP ESI

POP EBX

ADD [ESI], EAX ; X2: \_\_\_\_\_\_\_\_\_\_\_\_

ADD [EBX], EAX ; X1: \_\_\_\_\_\_\_\_\_\_\_\_

MOV ESP, EBP

POP EBP

RET 8 ; EIP: \_\_\_\_\_\_\_\_\_\_\_\_

P1 ENDP

**Part II**

Q. No 2 Answer all the questions in this section. [22x2=44]

.DATA

BARRAY BYTE 10H, 20H, 30H, 6 DUP (0AH)

ALIGN 4

WARRAY WORD 5 DUP(1000H)

PRESSKEY EQU <"PRESS ANY KEY TO CONTINUE ...",0>

DARRAY DWORD 5 DUP(56789ABH),7 DUP(12345678H)

PROMPT BYTE PRESSKEY

What will be the value of EAX, and AL after executing each of the following instructions? Assume that the address of barray is 404000h.

1. MOV EAX, TYPE WARRAY ; EAX =
2. MOV EAX, LENGTHOF BARRAY ; EAX =
3. MOV EAX, SIZEOF DARRAY ; EAX =
4. MOV EAX, OFFSET WARRAY ; EAX =
5. MOV EAX, DWORD PTR BARRAY ; EAX =
6. MOV AL, BYTE PTR DARRAY ; AL =
7. Would the following instruction set the zero flag? Explain.

MOV AX, 0000h ;clear the AX register

1. Is it possible for a NEG instruction to set the Overflow flag?

Consider a program that has the following data segment:

I EQU 2Eh, 2h

J BYTE '6789'

K EQU 140

L WORD 3412h, 8765h

M DWORD 4, 3, 5, 6, 7

Indicate whether the following instructions are valid or not. If valid, give the result of the operation in hexadecimal. If invalid, give the reason.

1. MOV AL, I+1

1. MOV AL, J+2
2. MOVSX EAX, L[1]
3. MOV EBX, M[2]
4. INC [ESI] ;ESI = OFFSET J
5. MOV I, L
6. MOV EAX, DWORD PTR J
7. MOV L, WORD PTR M
8. MOV ESI, L
9. Consider the following code:

mov ax, 0h

mov cx, 0Ah

doLoop:

dec ax

loop doLoop

What is the value of the *ax* register after the completion of the doLoop?

Ax =

1. When an interrupt occurs, arrange the following operations in their order of occurrence?

a) interrupt service routine executed

b) the registers are restored by popping their values off of the stack

c) the processor identifies the source of the interrupt

d) the program counter and other registers' values are pushed onto the stack

e) the address of the interrupt service routine is placed in the program counter

1. 2. 3. 4. 5. [02]
2. In the following code sequence, show the value of AL after each shift or rotate instruction has executed:

mov al,0D4h

shr al,1 ; a. AL =

mov al,0D4h

sar al,1 ; b. AL =

Suppose that you have the following initial register content: AX=F2E9H, BX=0002H CX=08A0H and DX=F1E0H

1. Show the contents of AX and the flags (CF,OF,SF and ZF) after executing:

ADD AX, BX ; a. CF = b. OF= c. SF= d. ZF=

1. Show the contents of CX and the flags (CF,OF,SF and ZF) after executing:

SUB CX, DX ; a. CF = b. OF= c. SF= d. ZF=

1. Show the contents of BX and the flags (CF,OF,SF and ZF) after executing:

NEG BX ; a. CF = b. OF= c. SF= d. ZF=

1. After the execution of the following sequence of instructions, what is the value of EAX?   
   MOV AH, 9Fh   
   MOV AL, FFh   
   XOR AH,AH   
   OR AH,AL

EAX =

1. Write a single instruction to mask out 1st and 3rd nibble of EAX.
2. Compares the integers 7FFFh and 8000h and show how the JB (unsigned) and JL (signed) instructions would generate different results.

Question No.3 : Assembly Language Programming [7x5=35 Points]

1. Implement the following pseudo-code in assembly language (Intel IA-32 and MIPS code) . Also, give the corresponding data definition directives:

(a)

; All values are

; 32-bit signed integers

while (OP1 < OP2)

{

OP1++;

if (OP3 == OP2)

X = Y + 2;

else

X = Y + 10;

}

(b)

; All values are

; 32-bit unsigned integers

if(VAL1>VAL2) AND (VAL2>VAL3) then

X=10

else

X=20

1. Write an assembly language procedure MINIMUM that is called from the MAIN procedure to find the minimum MIN among X, Y and Z. The arguments are passed by value to the procedure MINIMUM using registers. The result is also returned in a register. Also, write the corresponding data definition directives. The Intel IA 32 version of this program is required.
2. Suppose that there are two tables defined in the data segment, DS=2FF0H, namely Table1 and Table2. Table1 is at offset 1000H and Table2 is at offset 2000H. Both tables have a size of 100 bytes.

***Solve here***

1. Write a code segment to copy the content of Table1 to Table2.
2. Write a subroutine to search for a constant number that can be represented in a byte, in a table, and returns the index of the table where the number is found in the DI register. Assume that the constant number to be searched is pushed first in the stack, followed by the table address, and finally the size of the table. Then, write a code segment to search for the number 5 in Table1 and the number 10 in Table2, using the subroutine, and store the corresponding indices in registers AX and BX respectively.

(iv) Write an Assembly Language program to compute (a) the binomial coefficients C(n, k) and Power (X, N) using the recursive definition:

1. binomial coefficients C(n, k)
2. Power (X, N)

int Power(int X, int N) {

        if( N == 0 ) return 1;

        else return **Power( X, N-1)** \* X;

}

void main(void) {

        cout <<**Power(5,2)**;

}

1. Write an Assembly Language program to find the nth term Fibonacci Sequence:

|  |  |  |
| --- | --- | --- |
| 01 | int fibonacci(int n) | |
| 02 | { |

|  |  |  |
| --- | --- | --- |
| 03 | if(n==0) return0; | |
| 04 | else |

|  |  |
| --- | --- |
| 05 | if(n==1) return1; |
| 06 | elsereturnfibonacci(n - 1) + fibonacci(n - 2); | |

|  |  |  |
| --- | --- | --- |
| 07 | } | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | int main() | |
| 10 | { |

|  |  |
| --- | --- |
| 11 | int input; |
| 12 | cin >> input; | |

|  |  |  |
| --- | --- | --- |
| 13 | cout << fibonacci(input) << endl; | |
| 14 | } |

(vi) **EXCHANGE SORT**

The exchange sort is similar to its cousin, the bubble sort, in that it compares elements of the array and swaps those that are not in their proper positions.  (Some people refer to the "exchange sort" as a "bubble sort".)  The difference between these two sorts is the manner in which they compare the elements. The exchange sort compares the first element with each following element of the array, making any necessary swaps.

for (i = 0; i < n-1; i++)

for (j = 0; j < n-i-1; j++)

if (a[j] > a[j+1])

{

t = a[j];

a[j] = a[j+1];

a[j+1] = t;

}

Write an assembly Language program to sort the elements using exchange sort.

**(vii) SELECTION SORT**

Selection sort carries out a sequence of passes over the table. At the first pass an entry is selected on some criteria and placed in the correct position in the table. The possible criteria for selecting an element are to pick the smallest or pick the largest. If the smallest is chosen then, for sorting in ascending order, the correct position to put it is at the beginning of the table. Now that the correct entry is in the first place in the table the process is repeated on the remaining entries. Once this has been repeated *n*-1 times the *n*-1 smallest entries are in the first *n*-1 places which leaves the largest element in the last place. Thus only *n*-1 passes are required. The algorithm can be described as follows:

for (i = 0; i < n-1; i++)

{

// find smallest entry in ith to n-1 th place

// p is subscript of smallest entry yet found

p = i;

for (j = i+1; j < n; j++)

if (a[j]<a[p])

p = j;

// exchange pth element with ith element

t = a[p];

a[p] = a[i];

a[i] = t;

}

For intimation, you can visit the below link:

Write an assembly Language program to sort all the elements using Selection sort.

**Part III**

Q. No. 4 Assembly Language + MIPS [5x9= 45 Points]

(i) Suppose the following data is received from a wireless sensor node operating in a smart building and is stored in EAX register, as shown in Figure 1. You are required to write an assembly language program in Intel IA 32 assembly with the corresponding data definition directives that would extract the data items and store them at memory locations Sequence\_Number, Revision\_Count, Status, and Sensor\_Data.

1. Bits 0 to 11 reflect an integer Sequence\_Number of the packet being sent.
2. Bits 12 – 14 show an integer Revision\_Count of the packet.
3. Bit 15 is the Status of the sensor flag (0 – Forwarded Data and 1 – Sensed Data)
4. Bits 16 – 31 contain the Sensor\_Data.

|  |  |  |  |
| --- | --- | --- | --- |
| 16 bits | 1 bit | 3 bits | 12 bits |
| Sensor\_Data | Status | Revision\_  Count | Sequence\_Number |
|  |  |  |  |

Figure: 1

1. Using shift and add instructions multiply a decimal number X10 by 2310. Assume that the result does not exceed the range of a16-bit register. The Intel IA 32 version of this program is required.

1. Give the contents of the following registers, along with the run-time stack, when the following instructions are executed. Initially, consider ESP = 00001FF8h.

Note: SOLVE THIS PART HERE. No Marks will be awarded without proper working using the stack diagrams.

X1 DWORD 25H

X2 DWORD 27H

MAIN PROC

PUSH 6H

PUSH 5H

CALL P1

11500000H MOV RESULT, EAX ; ESP: \_\_\_\_\_\_\_\_\_\_\_\_

MAIN ENDP

P1 PROC

115000A4H PUSH EBP

MOV EBP, ESP ; EBP: \_\_\_\_\_\_\_\_\_\_\_\_

MOV EAX, [EBP+8]

ADD EAX, [EBP+12] ; EAX: \_\_\_\_\_\_\_\_\_\_\_\_

PUSH OFFSET X1

PUSH OFFSET X2 ; ESP: \_\_\_\_\_\_\_\_\_\_\_\_

POP ESI

POP EBX

ADD [ESI], EAX ; X2: \_\_\_\_\_\_\_\_\_\_\_\_

ADD [EBX], EAX ; X1: \_\_\_\_\_\_\_\_\_\_\_\_

MOV ESP, EBP

POP EBP

RET 8 ; EIP: \_\_\_\_\_\_\_\_\_\_\_\_

P1 ENDP

1. Write an assembly language program to copy the characters of a string to a target string. The characters are stored in such a way that only a single instance of any character in the string is stored. Initialize a source string to: "This is the source string".
2. Write an ASM program that prompts the user to enter a string of at most 128 characters and prints it in reverse order with every upper-case letter converted to the corresponding lower-case letter.

Enter a string of at most 128 characters:

An InpuT Line!

Here it is in LOWERCASE and in reverse order:

!enil tupni na

HINT: You should copy the user input line into memory (possibly after converting into lower-case) and then print it in reverse order by using indirect addressing.

If the user enters more than 128 characters, only the first 128 characters must be processed and the rest are ignored.