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

KARACHI CAMPUS



Grand AssignmentFall 2018

*Computer Organization & Assembly Language* (EE 213)

Total Points: **155**

Solve on this paper, and attached the program results

Roll No: 17k-3730 Section :C Signature: \_\_\_\_\_\_\_\_\_\_

Question No. 1: Programming Basics [10\*02 = 20 Points]

Machine Language

(i) The following bytes are found in order somewhere in memory. Assuming they are machine codes, decode the values into meaningful assembly language mnemonics.

B9 00 12,8C 85 DC 01

a)mov ax,1200h

b)mov [di+01DCh],al

(ii) Convert the following independent Assembly Language instructions to Machine Language code – give your answers in hexadecimal:

MOV [SI+490], SP

1000100110100100 -> 89A40049

ADD AL, [BX + SI]

0000001000000000 -> 0200

JNZ NEXT ; NEXT is a label at offset 0008H and

750800

PUSH AX

50+0 -> 50

MOV AX, VAR + 6 ; OFFSET of VAR is 0002H

A1h

SUB CX, VAR2 ; OFFSET of VAR2 is 0008H

2B0Eh

INC DX

40+2 -> 42

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

MOV AL,7AH

NOT AL ; a. AL =85

MOV AL,3DH

AND AL, 74H ; b. AL =34

MOV AL,9BH

OR AL,35H ; c. AL =BF

MOV AL,72H

XOR AL,0DCH ; d. AL =AE

(iv) Differentiate between the following Assembly Language instructions:

a) MOV EAX, OFFSET VAR1

b) MOV EAX, VAR1

a) Offset of variable is stored in eax

b) Value of var1 in stored in eax.

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

* Temporary storage area for registers.
* Used for passing parameters during a procedure call.
* Allocating local variables used inside procedures.
* Keep return addresses.
* It supports recursion.

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

PUSH EAX

PUSH EBX

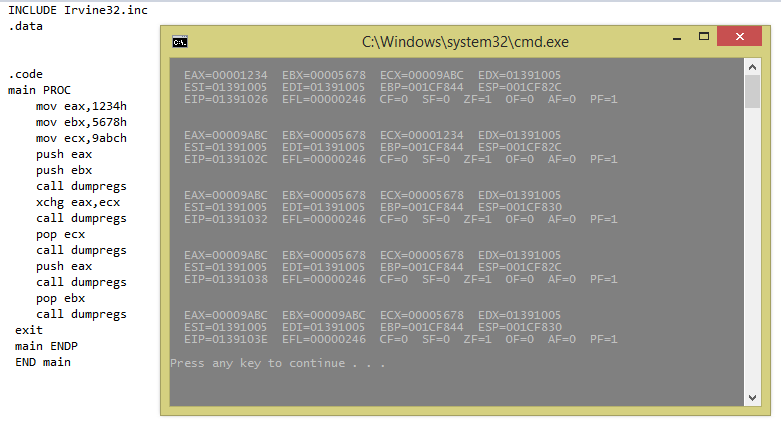
XCHG EAX, ECX

POP ECX

PUSH EAX

POP EBX

1. EAX :9ABCh
2. EBX :9ABCh ECX :5678h
3. ESP : 00FCh



(vii) 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

MYSUM PROC

Push ESI

Push ECX

MOV ECX,10

L1:

ADD EAX, [ESI]

SUB ESI, 4

LOOP L1

POP ECX

POP ESI

Ret

MYSUM ENDP

(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.

**----------------------------------**

**CS=508A0h 100h**

**Code Segment**

**-----------------------------------**

**DS=507A0h 50h**

**Data Segment**

**-----------------------------------**

**SS=505A0h 200h**

**Stack Segment**

**-----------------------------------**

(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 | X | X | yes |
| If V1<V2 then | x | Yes | x |
| If V1>V2 then | yes | x | X |

V1DB(?)

V2DB(?)

.CODE

Start:

•

•

MOV AL, V1

CMP AL, V2

JZ Label1

JS Label1

MOVAH, 1

JMPContinue

Label1:

JELabel2

MOVAH, 2

JMPContinue

Label2:

MOVAH, 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: 00001FF8h

MAIN ENDP

P1 PROC

115000A4H PUSH EBP

MOV EBP, ESP ; EBP: 1FECh

MOV EAX, [EBP+8]

ADD EAX, [EBP+12] ; EAX: Bh

PUSH OFFSET X1

PUSH OFFSET X2 ; ESP: 1FE4h

POP ESI

POP EBX

ADD [ESI], EAX ; X2: 32h

ADD [EBX], EAX ; X1: 30h

MOV ESP, EBP

POP EBP

RET 8 ; EIP: 11500000h

P1 ENDP

Q. No 2 Answer all the questions in this section. [2x22=

.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
2. MOV EAX, LENGTHOF BARRAY ; EAX = 9
3. MOV EAX, SIZEOF DARRAY ; EAX = 30
4. MOV EAX, OFFSET WARRAY ; EAX = 40400Ch
5. MOV EAX, DWORD PTR BARRAY ; EAX = 0A302010h
6. MOV AL, BYTE PTR DARRAY ; AL = ‘<’
7. Would the following instruction set the zero flag? Explain.

MOV AX, 0000h ;clear the AX register

**Yes**

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

**Yes**

Consider a program that has the following data segment:

I EQU 2Eh

J BYTE '6789'

K EQU 140

L WORD 3412h, 8765h

M DWORD 4, 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

Al=002F

1. MOV AL, J+2

Al=08

1. MOVSX EAX, L[1]

Eax=00006534h

1. MOV EBX, M[2]

Ebx=00050000h

1. INC [ESI] ;ESI = OFFSET J

Not valid-> operand must have same size

1. MOV I, L

Not valid-> source and destination both are variables.

1. MOV EAX, DWORD PTR J

Eax=39383736h

1. MOV L, WORD PTR M

Not valid-> both are memory operands.

1. MOV ESI, L

Not valid-> operands must have same size.

1. Consider the following code:

mov ax, 0h

mov cx, 0Ah

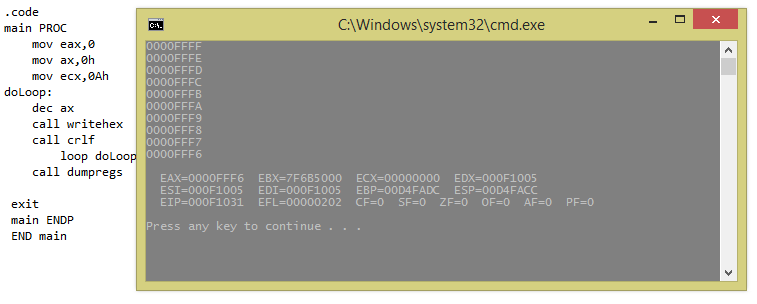
doLoop:

dec ax

loop doLoop

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

Ax=0000FFF6



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

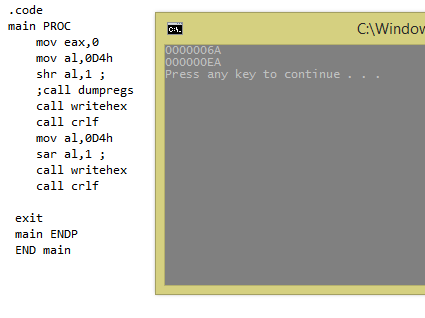
1. c 2. d 3. e 4. a 5. b [02]
2. In the following code sequence, show the value of AL after each shift or rotate instruction has executed:

mov al,0D4h

shr al,1 ; a. AL = 0000006Ah

mov al,0D4h

sar al,1 ; b. AL = 000000EAh



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 =0 b. OF=1 c. SF=1 d. ZF=0

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

SUB CX, DX ; a. CF =1 b. OF=0 c. SF=0 d. ZF=0

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

NEG BX ; a.CF =1 b.OF=0 c.SF=1 d.ZF=0

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 = 0000FFFFh

1. Write a single instruction to mask out 1st and 3rd nibble of EAX.

AND eax,0000111100001111

1. Compares the integers 7FFFh and 8000h and show how the JB (unsigned) and JL (signed) instructions would generate different results.

Both will generate same result

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++;

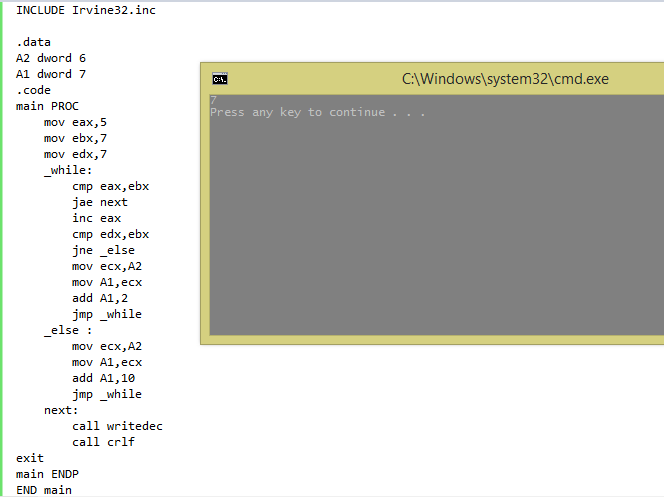
if (OP3 == OP2)

X = Y + 2;

else

X = Y + 10;

}



(b)

; All values are

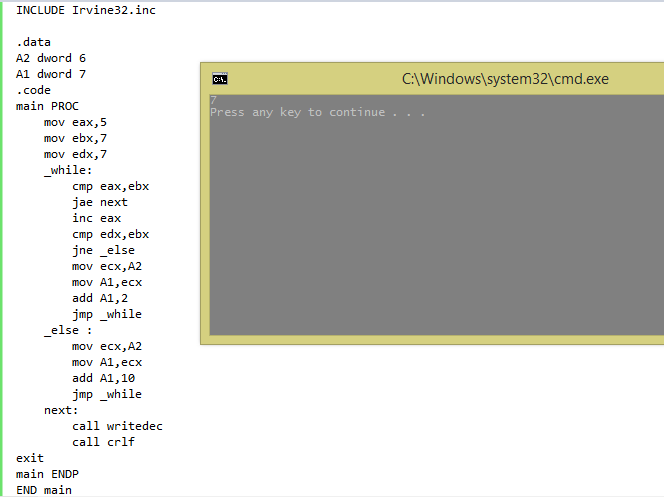
; 32-bit unsigned integers

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 and MIPS version of this program is required.

INCLUDE Irvine32.inc

.data

var1 Dword 5

var2 dword 2

var3 dword 1

X dword ?

msg byte "Minimum value is : ",0

.code

main PROC

mov eax,var1

mov ebx,var2

mov edx,var3

call Minimun

exit

main ENDP

Minimun proc

cmp eax,ebx

jb Checksecond

jmp Checkthird

Checksecond:

cmp eax,edx

ja Checkthird

mov X,eax

Checkthird:

cmp ebx,edx

ja l1

mov X,ebx

l1:

mov X,edx

mov eax,X

mov edx,offset msg

call writestring

call writedec

call crlf

ret

Minimun endp

END main



1. 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.

INCLUDE Irvine32.inc

.data

table1 BYTE 2,3,1,9,0,5,6,10,2,9,1,89 DUP(0)

table2 BYTE 100 DUP(0)

msg1 byte "Index of 5 : ",0

msg2 byte "Index of 10 : ",0

.code

main PROC

mov edi,offset table1

mov esi,offset table2

mov eax,0

mov ecx,lengthof table1

l1:

mov al, [edi]

mov [esi],al

add edi,1

add esi,1

loop l1

push 5 ;to be searched

push OFFSET table1

push 100

call Firstsearch

call crlf

push 10 ;to be searched

push OFFSET table1

push 100

call Secondsearch

exit

main ENDP

Firstsearch PROC

push ebp

mov ebp , esp

mov al , [ebp+16]

mov esi , [ebp+12]

mov ecx , [ebp+8]

mov edx ,LENGTHOF table1

X1:

mov bl,[esi]

cmp bl , al

je X2

inc esi

Loop X1

jmp quit

X2:

sub edx , ecx

mov eax , edx

mov edx,offset msg1

call writestring

call WriteDec

jmp quit

quit:

pop ebp

ret 16

Firstsearch ENDP

Secondsearch PROC

push ebp

mov ebp , esp

mov al , [ebp+16]

mov esi , [ebp+12]

mov ecx , [ebp+8]

mov edx ,LENGTHOF table1

X1:

mov bl,[esi]

cmp bl , al

je X2

inc esi

Loop X1

jmp quit

X2:

sub edx , ecx

mov eax , edx

mov edx,offset msg2

call writestring

call WriteDec

call crlf

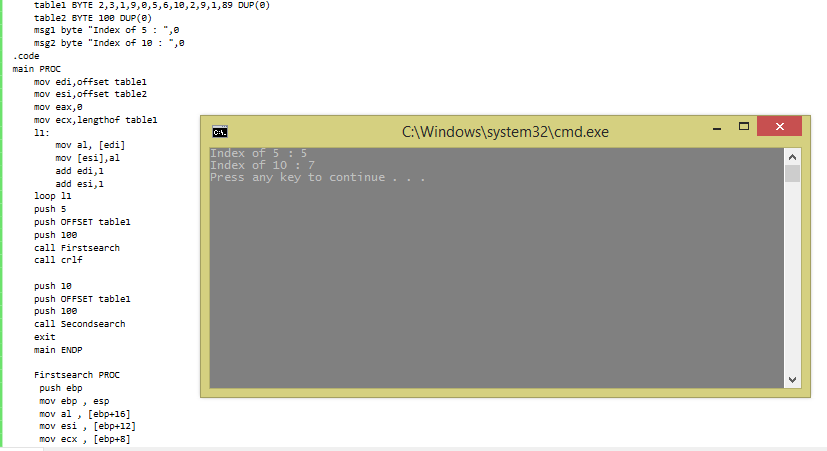
jmp quit

quit: pop ebp

ret 16

Secondsearch ENDP

END main



(iv) Write anAssembly Languageprogram 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)**;

}

INCLUDE Irvine32.inc

.data

temp DWORD ?

X DWORD ?

N DWORD ?

msg byte "power : ",0

.code

main PROC

mov eax,0

call readdec

mov X,eax

mov temp,eax

call readdec

mov N,eax

mov ebx,eax

push X

push N

call power

exit

main ENDP

power proc

push ebp

mov ebp,esp

mov eax,[ebp+12]

mov ebx,[ebp+8]

cmp ebx,1

jne l1

jmp quit

l1:

mul temp

dec ebx

push eax

push ebx

call power

quit: mov edx,offset msg

call writestring

call writedec

call crlf

ret 8

power endp

END main



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

|  |  |  |
| --- | --- | --- |
| 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 | } |

INCLUDE IRVINE32.inc

.data

msg1 byte "The n terms of sequence are : ",0

N dWORD ?

.code

main PROC

mov eax,0

call readint

mov N,eax

mov al,0

mov bl,1

mov eax,0

mov edx,offset msg1

call writestring

call crlf

call writedec

mov al,bl

call crlf

call writedec

call crlf

sub N,2

mov ecx,N

mov edx,0

l1:

add dl,al

mov al,dl

call writedec

call crlf

mov al,bl

mov bl,dl

loop l1

exit

main ENDP

END main



1. **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.

Include Irvine32.inc

exchange\_sort proto ,pointerarray:ptr sdword,count:dword

.data

msg1 byte "Array before exchange sort is : ",0

msg2 byte "Array after exchange sort is : ",0

array sdword 3,1,7,5,2,-9,4,3

arraysize dword 8

.code

main PROC

mov edx,offset msg1

call writestring

call crlf

mov ecx,arraysize

mov edi,offset array

l1:

mov eax,[edi]

call writeint

call crlf

add edi,4

loop l1

call crlf

invoke exchange\_sort ,addr array,arraysize

mov edx,offset msg2

call writestring

call crlf

mov ecx,arraysize

mov edi,offset array

l2:

mov eax,[edi]

call writeint

call crlf

add edi,4

loop l2

call crlf

exit

main ENDP

exchange\_sort proc ,pointerarray:ptr sdword ,count:dword

mov ecx,count

dec ecx

l3:

push ecx

mov edi,pointerarray

l4:

mov eax,[edi]

cmp [edi+4],eax

jg l5

xchg eax,[edi+4]

mov [edi],eax

l5 :

add edi,4

loop l4

pop ecx

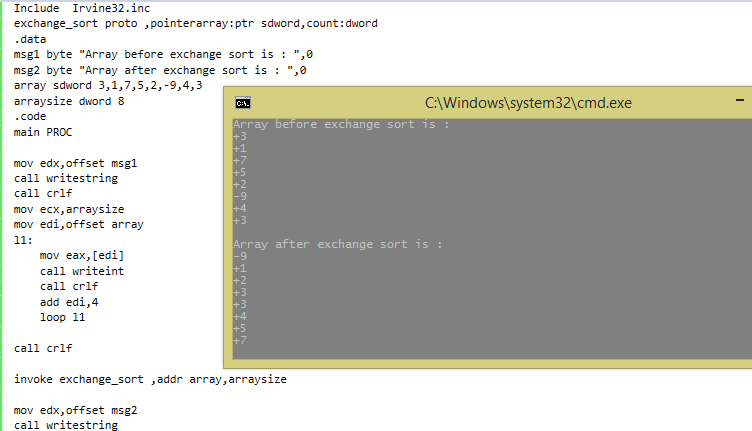
loop l3

l6:

ret

exchange\_sort endp

END main



**(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.

Include Irvine32.inc

\_sort proto ,pointerarray:ptr sdword,count:dword

.data

msg1 byte "Array before exchange sort is : ",0

msg2 byte "Array after exchange sort is : ",0

array sdword 3,1,7,5,2,-9,4,3

arraysize dword 8

.code

main PROC

mov edx,offset msg1

call writestring

call crlf

mov ecx,arraysize

mov edi,offset array

l1:

mov eax,[edi]

call writeint

call crlf

add edi,4

loop l1

call crlf

invoke \_sort ,addr array,arraysize

mov edx,offset msg2

call writestring

call crlf

mov ecx,arraysize

mov edi,offset array

l2:

mov eax,[edi]

call writeint

call crlf

add edi,4

loop l2

call crlf

exit

main ENDP

\_sort proc ,pointerarray:ptr sdword ,count:dword

mov ecx,count

dec ecx

l3:

push ecx

mov edi,pointerarray

l4:

mov eax,[edi]

cmp [edi+4],eax

jg l5

xchg eax,[edi+4]

mov [edi],eax

l5 :

add edi,4

loop l4

pop ecx

loop l3

l6:

ret

\_sort endp

END main

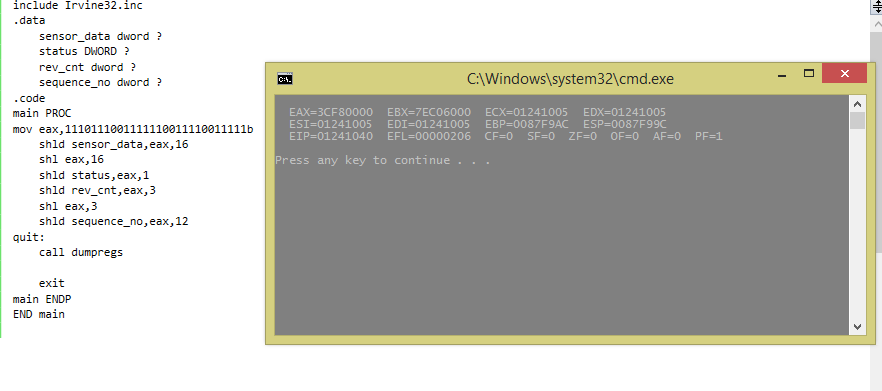
Q. No. 4 Assembly Language + MIPS [9x5= 45 Points]

(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 (a) Intel IA 32 and (b) in MIPS assemblywith 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



include Irvine32.inc

.data

sensor\_data dword ?

status DWORD ?

rev\_cnt dword ?

sequence\_no dword ?

.code

main PROC

mov eax,1110111001111110011110011111b

shld sensor\_data,eax,16

shl eax,16

shld status,eax,1

shld rev\_cnt,eax,3

shl eax,3

shld sequence\_no,eax,12

quit:

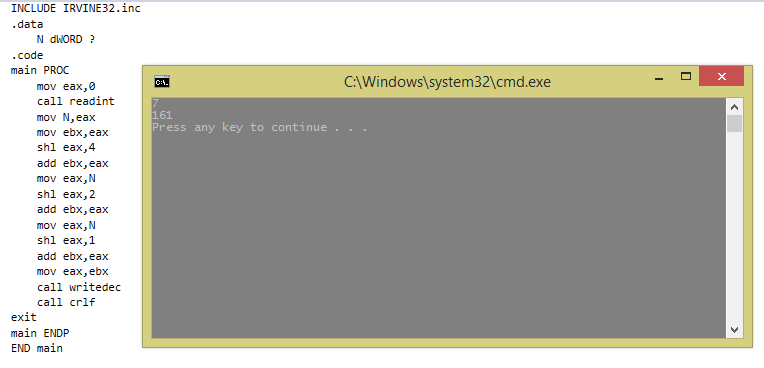
call dumpregs

exit

main ENDP

END main

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 and MIPS version of this program is required.



INCLUDE IRVINE32.inc

.data

N dWORD ?

.code

main PROC

mov eax,0

call readint

mov N,eax

mov ebx,eax

shl eax,4

add ebx,eax

mov eax,N

shl eax,2

add ebx,eax

mov eax,N

shl eax,1

add ebx,eax

mov eax,ebx

call writedec

call crlf

exit

main ENDP

END main

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:00001ff8h

MAIN ENDP

P1 PROC

115000A4H PUSH EBP

MOV EBP, ESP ; EBP: 1FECh

MOV EAX, [EBP+8]

ADD EAX, [EBP+12] ; EAX: Bh

PUSH OFFSET X1

PUSH OFFSET X2 ; ESP: 1fe4h

POP ESI

POP EBX

ADD [ESI], EAX ; X2: 32h

ADD [EBX], EAX ; X1: 30h

MOV ESP, EBP

POP EBP

RET 8 ; EIP: 11500000h

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".

INCLUDE IRVINE32.inc

.data

source byte "This is the source string",0

dest byte 50 dup(?)

.code

main PROC

mov ecx,lengthof source

mov esi,offset source

mov edi,offset dest

rep movsb

mov edx,offset dest

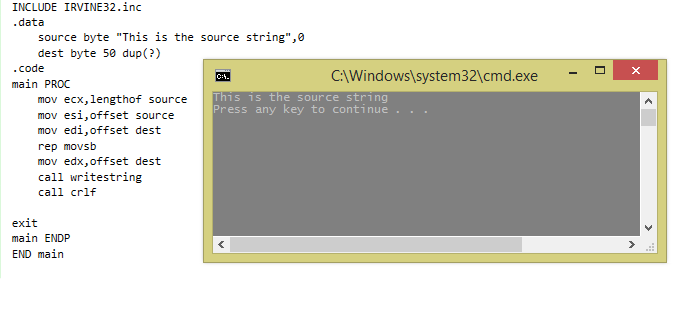
call writestring

call crlf

exit

main ENDP

END main



1. Write a recursive procedure to find a value in a large integer array. Ask the user to enter an integer value in the main program. You should pass user supplied value as parameter to the recursive function using the INVOKE directive. Also, draw labeled diagrams to show stack values at each iteration of this recursive function.

include Irvine32.inc

.data

array DWORD 1,2,3,4,5,6,7,8,9,10

msg BYTE "not found",0

msg1 BYTE "found",0

.code

recursiveSearch proto,parray:PTR DWORD,val:DWORD

main PROC

INVOKE recursiveSearch,addr array,2

l1:

call dumpregs

exit

main ENDP

recursiveSearch PROC, parray:PTR DWORD,val:DWORD

push ebp

mov ebp,esp

mov esi ,parray

mov eax ,val

L1:

cmp eax,[esi]

je L2

add esi,4

loop L1

mov edx,offset msg1

call writestring

jmp t

L2:

mov edx , offset msg

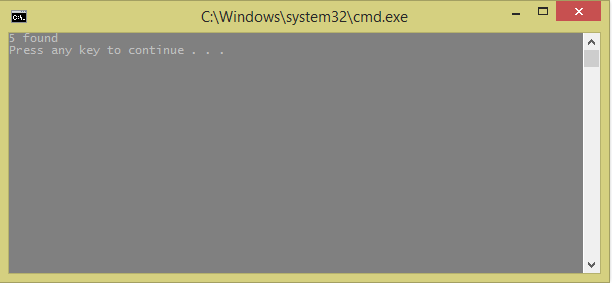
call writestring

t:

ret

recursiveSearch endp

END main



1. Write an assembly language code to implement the following high-level language code showing the use of LEA instruction and OFFSET assembler directive.

char moon [20];

void star\_array () {

char cell[20];

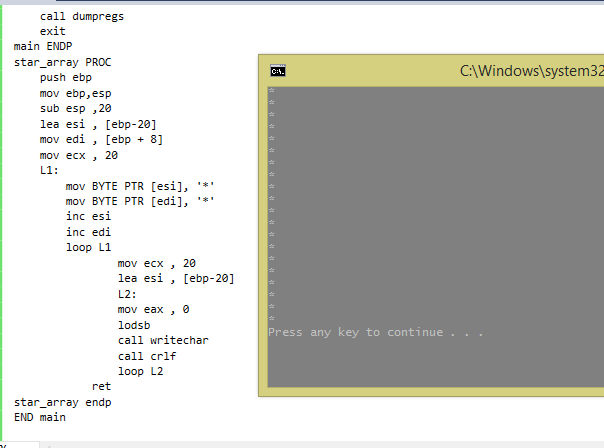
for (int i=19; i>=0; i--) {

cell[i] = ‘\*’;

moon[i] = ‘x’;

}

}



include Irvine32.inc

.data

moon BYTE 20 DUP(?)

.code

main PROC

push offset moon

call star\_array

ext:

call dumpregs

exit

main ENDP

star\_array PROC

push ebp

mov ebp,esp

sub esp ,20

lea esi , [ebp-20]

mov edi , [ebp + 8]

mov ecx , 20

L1:

mov BYTE PTR [esi], 'x'

mov BYTE PTR [edi], '\*'

inc esi

inc edi

loop L1

mov ecx , 20

lea esi , [ebp-20]

L2:

mov eax , 0

lodsb

call writechar

call crlf

loop L2

ret

star\_arrayendp

END main

1. Write a recursive procedure in x86 assembly language that divides a number by another number and stops when dividend is less than or equal to 5h. Consider dividend = D4A4h and divisor = Ah.The Intel IA 32 and MIPS version of this program is required. In MIPS Assembly you have a choice to use the simple loop-based implementation.



include Irvine32.inc

.data

.code

main PROC

push 0D4A4h

push 0Ah

call divProc

ext:

call dumpregs

exit

main ENDP

divProc proc

push ebp

mov ebp,esp

mov eax,[ebp + 12]

mov ebx,[ebp + 8]

div ebx

cmp eax,5h

jle mylabel

push eax

push ebx

call divProc

call writedec

mylabel:

mov esp,ebp

pop ebp

ret 8

divProc endp

END main

viii) Using string primitives, write an assembly language program that searches 20 elements of array ArraySearchValues in 1000 un sorted elements of another array ArrayValues.

include Irvine32.inc

.data

unArray DWORD 100 dup(8)

search DWORD 20 dup(2)

msg BYTE "not found",0

msg1 BYTE "found",0

count DWORD ?

.code

main PROC

mov edi,offset unArray

mov esi,offset search

mov ecx,20

cld

L1:

mov edi,offset unArray

mov eax,[esi]

repe scasd

jnz L3

mov edx,offset msg1

call writestring

L4:

add esi,4

loop L1

jmp quit

L3:

mov edx,offset msg

call writestring

mov eax,[esi]

call writedec

call crlf

jmp L4

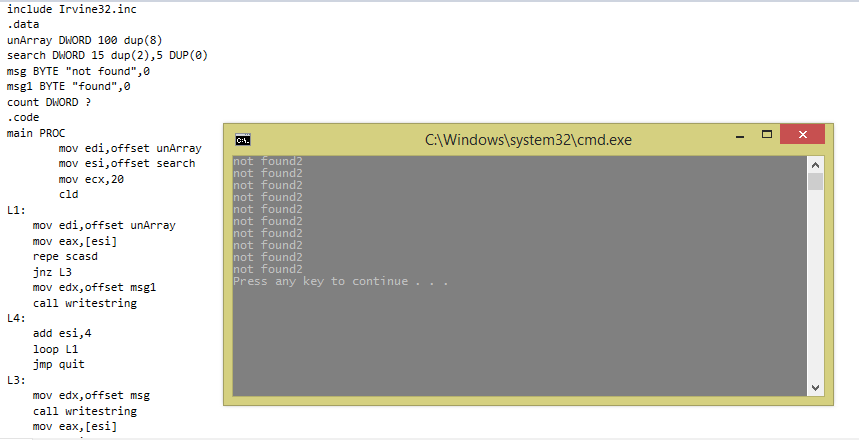
quit:

;call dumpregs

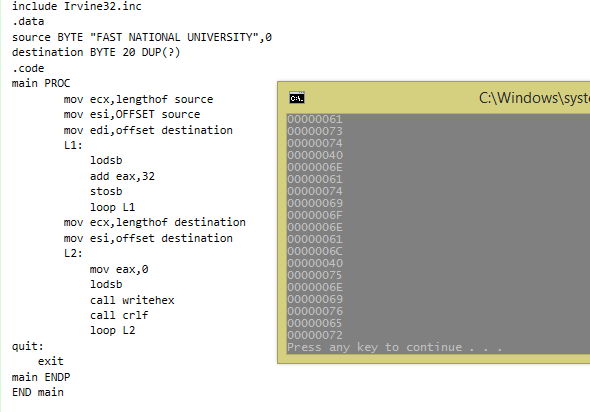
exit

main ENDP

END main



1. Using string primitives, write a program that converts the string “FAST NATIONAL UNIVERSITY” to its respective ASCII values into a new array. Also, write a procedure to search a particular string SITYA defined in the data directives.



include Irvine32.inc

.data

source BYTE "FAST NATIONAL UNIVERSITY",0

destination BYTE 20 DUP(?)

.code

main PROC

mov ecx,lengthof source

mov esi,OFFSET source

mov edi,offset destination

L1:

lodsb

add eax,32

stosb

loop L1

mov ecx,lengthof destination

mov esi,offset destination

L2:

mov eax,0

lodsb

call writehex

call crlf

loop L2

quit:

exit

main ENDP

END main