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

KARACHI CAMPUS



Grand Assignment Fall 2018

*Computer Organization & Assembly Language* (EE 213)

Total Points: **155**

Solve on this paper, and attached the program results

Roll No: 18k-1259 Section:**D** 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

**ANSWER:**

* **B9 00 12**

**B8+regd =** MOV REG 16; IMM 16

**SO, B9 = M**OV CX, 1200h

|  |
| --- |
| **MOV CX, 1200h** |

* **8C 85 DC 01**

**8C =** MOV REG16/ MEM16; SREG

**2ND BYTE = MOD 0 SR R/M**

**85 =** 10 0 00 101

**MOD =** 10

**SR =** 00**(ES)**

**R/M =** 101 **[(DI) + D16]**

**SO,** MOV [DI]+01DCh, ES

|  |
| --- |
| **MOV [DI]+01DCh, ES** |

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

**ANSWER:**

* MOV [SI+490], SP

**490d =** 01EAh

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **OPCODE** | **D** | **W** | **MOD** | **REG** | **R/M** | **LOW DES** | **HIGH DES** |
| 100010 | 0 | 1 | 10 | 100 | 100 | EA | O1 |
| **1000 1001 1010 0100 => 89 Ach** | | | | **EA 01h** | | | |
| **89 AC EA 01h** | | | | | | | |

* ADD AL, [BX + SI]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **OPCODE** | **D** | **W** | **MOD** | **REG** | **R/M** |
| 000000 | 1 | 0 | 00 | 000 | 000 |
| **0000 0010 0000 0000 => 02 00h** | | | | | |

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

**JNZ 0008h =>** 75 08 00

|  |
| --- |
| **75 08 00h** |

* PUSH AX

**PUSH REG16 =>**11111111 MOD 110 R/M

**MOD =>**11

**R/M =>**000

**SO, PUSH AX =>**11111111 11 110 000

|  |
| --- |
| **1111 1111 1111 0000 => FF F0h** |

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

**DIRECT ADDRESS => VAR+06h =>**0002h+0006h **=>**0008h

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **OPCODE** | **D** | **W** | **MOD** | **REG** | **R/M** |
| 100010 | 1 | 1 | 00 | 000 | 110 |
| **1000 1011 0000 0110 =>8A 06h** | | | | | |
| **8A 06 08 00h** | | | | | |

* SUB CX, VAR2 ; OFFSET of VAR2 is 0008H

**DIRECT ADDRESS =>**0008h

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **OPCODE** | **D** | **W** | **MOD** | **REG** | **R/M** |
| 001010 | 1 | 1 | 00 | 001 | 110 |
| **0010 1011 0000 1110 => 2A 0Eh** | | | | | |
| **2A OE 08 00h** | | | | | |

* INC DX

**INC REG16 =>**40 +REGCD

**REG => 010**

**SO, PUSH DX =>**40 + 2 **=> 42h**

|  |
| --- |
| **42h** |

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

**ANSWER:**

MOV AL,7AH

NOT AL ; a. AL =**85h**

MOV AL,3DH

AND AL, 74H ; b. AL = **34h**

MOV AL,9BH

OR AL,35H ; c. AL = **BFh**

MOV AL,72H

XOR AL,0DCH ; d. AL = **AEh**

(iv) Differentiate between the following Assembly Language instructions:

MOV EAX, OFFSET VAR1

MOV EAX, VAR1

**ANSWER:**

1. **MOV EAX, OFFSET VAR1**

In this command, the address of **VAR1** will store in **EAX.**

**Example:**

; 0X000AE VAR1 = 10

MOV EAX, OFFSET VAR1

; EAX = 0X000AE

1. **MOV EAX, VAR1**

In this command, the value of **VAR1** will store in **EAX.**

**Example:**

; 0X000AE VAR1 = 10

MOV EAX, VAR1

; EAX = 10

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

**ANSWER:**

1. To store values temporarily.
2. To use local variables.
3. To store return address of PROCEDURES.
4. To use 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 the following instructions:

PUSH EAX

PUSH EBX

XCHG EAX, ECX

POP ECX

PUSH EAX

POP EBX

**ANSWER:**

1. EAX => **9ABCh**
2. EBX =>**9ABCh**
3. ECX =>**5678h**
4. ESP =>**0FCh**

(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

**ANSWER:**

INCLUDE IRVINE32.INC

.DATA

ARR DWORD 1, 2, 3, 4, 5, 6, 7, 8, 9, 0

SUM DWORD 0

.CODE

MAIN PROC

MOV ECX, LENGTHOF ARR

MOV EAX, SIZEOF ARR

SUB EAX, TYPE ARR

MOV ESI, OFFSET ARR

ADD ESI, EAX

MOV EAX, 0

CALL MYSUM

MOV SUM, EAX

CALL WRITEDEC

CALL CRLF

EXIT

MAIN ENDP

MYSUM PROC USES ESI ECX

MOV ECX, 10

L1:

ADD EAX, [ESI]

SUB ESI, 4

LOOP L1

ret

MYSUM ENDP

END MAIN

(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.

**ANSWER:**

(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 | √ |  |  |

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: 00001FEC

|  |  |
| --- | --- |
| STACK | |
| 00001FE4h | X2 |
| 00001FE8h | X1 |
| 00001FECh | EBP |
| 00001FF0h | 11500000H |
| 00001FF4h | 5H |
| 00001FF8h | 6H |

MOV EAX, [EBP+8]

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

PUSH OFFSET X1

PUSH OFFSET X2 ; ESP: 00001FE0

POP ESI

POP EBX

ADD [ESI], EAX ; X2: 32H

ADD [EBX], EAX ; X1: 30H

MOV ESP, EBP

POP EBP

RET 8 ; EIP: 111500000H

P1 ENDP

Q. No 2 Answer all the questions in this section. [2x22=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**
2. MOV EAX, LENGTHOF BARRAY ; EAX = **9**
3. MOV EAX, SIZEOF DARRAY ; EAX = **48d**
4. MOV EAX, OFFSET WARRAY ; EAX = **40400DH**
5. MOV EAX, DWORD PTR BARRAY ; EAX = **0A302010H**
6. MOV AL, BYTE PTR DARRAY ; AL = **ABH**
7. Would the following instruction set the zero flag? Explain.

MOV AX, 0000h ;clear the AX register

**ANSWER:**

When the value 0000h moves in AX, then the value of AX becomes 0, and when the value of regiter becomes 0, the zero flag sets automatically.

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

**ANSWER:**

Yes, it is possible.

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.

**ANSWER:**

1. MOV AL, I+1

**AL = 2FH**

1. MOV AL, J+2

**AL = 38H**

1. MOVSX EAX, L[1]

**EAX = 6543H**

1. MOV EBX, M[2]

**EBX = 00050000H**

1. INC [ESI] ;ESI = OFFSET J

**It will not work, the size of register ESI should be same to the size of J.**

1. MOV I, L

**It will not work, immediate operands are not allowed.**

1. MOV EAX, DWORD PTR J

**EAX = 39383736H**

1. MOV L, WORD PTR M

**INVALID INSTRUCTION OPERAND.**

1. MOV ESI, L

**INSTRUCTION OPERANDS MUST BE THE SAME SIZE.**

1. Consider the following code:

mov ax, 0h

mov cx, 0Ah

doLoop:

dec ax

loopdoLoop

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

**ANSWER:**

**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 [02]

**ANSWER:**

1. **c**
2. **d**
3. **e**
4. **a**
5. **b**
6. 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 =6AH**

mov al,0D4h

sar al,1 ;**b. AL = 75H**

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

AX = **F2EBH**

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

CX = **16C0H**

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

BX =**FFFEH**

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.

**ANSWER:**

**MOV EAX, 43BF49AC H**

**; 1ST NIBBLE = C**

**; 3RDNIBBLE = 9**

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

**ANSWER:**

INCLUDE Irvine32.inc

.data

X SDWORD 7FFFH

Y SDWORD 8000H

.code

main proc

MOV EAX, X

CMP EAX, Y

JB UNSIGNED

CMP EAX, Y

JL SIGNED

JMP EN

UNSIGNED:

MOV EAX, 1

CALL DUMPREGS

CMP EAX, Y

JL SIGNED

JMP EN

SIGNED:

MOV EAX, 0

EN:

CALL DUMPREGS

exit

main ENDP

end main

**IN THIS CODE, THERE IS A DIFFERENCE OF PARITY BIT ONLY.**

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)

|  |
| --- |
| **(a):**  **INTEL IA-32**  **CODE:**  INCLUDE Irvine32.inc  .data  m byte "op1 = ", 0  op1 DWORD 20  op2 DWORD 26  op3 DWORD 22  x DWORD 0  y DWORD 2  .code  main proc  mov eax, op1  cmp eax, op2  jae quit  w:  inc eax  mov ebx, op3  cmp ebx, op2  je equal  mov ecx, y  add ecx, 10  mov x, ecx  jmp e  equal:  mov ecx, y  add ecx, 2  mov x, ecx  e:  cmp eax, op2  jb w  quit:  mov edx, OFFSET m  Call WriteString  Call WRITEDEC  CALL CRLF  exit  main ENDP  end main |

; All values are

; 32-bit signed integers

while (OP1 < OP2)

{

OP1++;

if (OP3 == OP2)

X = Y + 2;

else

X = Y + 10;

}

|  |
| --- |
| **(b):**  **INTEL IA-32**  **CODE:**  INCLUDE Irvine32.inc  .data  m byte "x = ", 0  val1 DWORD 26  val2 DWORD 24  val3 DWORD 20  x DWORD 0  .code  main proc  mov eax, val1  cmp eax, val2  jbe else\_  mov ebx, val2  cmp ebx, val3  jbe else\_  mov ecx, 10d  mov x, ecx  jmp quit  else\_:  mov ecx, 20d  mov x, ecx  quit:  mov eax, x  mov edx, OFFSET m  Call WriteString  Call WRITEDEC  CALL CRLF  exit  main ENDP  end main |

(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.

**ANSWER:**

**CODE IA-32:**

INCLUDE Irvine32.inc

.data

msg byte "MINIMUM = ",0

MIN dword ?

x DWORD 0Ch

y DWORD 0Bh

z DWORD 0Ah

.code

main PROC

mov eax,x

mov ebx,y

mov ecx,z

CALL MINIMUM

Mov edx, OFFSET msg

CALL writestring

CALL writedec

CALL CRLF

exit

main ENDP

MINIMUM proc

cmp eax,ebx

jb x\_

jmp y\_

x\_:

cmp eax,ecx

ja y\_

mov MIN,eax

y\_:

cmp ebx,ecx

ja z\_

mov MIN,ebx

z\_:

mov MIN,ecx

mov eax, MIN

ret

MINIMUM 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.

**ANSWER:**

INCLUDE Irvine32.inc

.data

m1 byte "THE POSITION OF CONSTANT in TABLE1 IS ",0

m2 byte "THE POSITION OF CONSTANT in TABLE2 IS ",0

Table1 BYTE 2, 5, 3, 6, 3, 10, 78, 3, 4, 6, 90 DUP (?)

Table2 BYTE 100 DUP (?)

.code

main PROC

mov esi,OFFSET Table1

mov edi,OFFSET Table2

mov ecx, LENGTHOF Table1

cld

rep MOVSB

push 5

push OFFSET Table1

push 100

Call s1

mov edx, OFFSET m1

CALL WRITESTRING

CALL WRITEDEC

CALL CRLF

push 10

push OFFSET Table2

push 100

Call s2

Move dx, OFFSET m2

CALL WRITESTRING

CALL WRITEDEC

CALL CRLF

exit

main ENDP

s1 PROC

push ebp

mov ebp , esp

mov al , [ebp+16]

mov esi , [ebp+12]

mov ecx , [ebp+8]

mov edx ,LENGTHOF table1

l1:

cmp al, [esi]

je equal

add esi, 1

loop l1

jmped

equal:

sub edx, ecx

mov eax, edx

ed:

pop ebp

add esi, 16

ret

s1 ENDP

s2 PROC

push ebp

mov ebp , esp

mov al , [ebp+16]

mov esi , [ebp+12]

mov ecx , [ebp+8]

mov edx ,LENGTHOF table2

l1:

cmp al, [esi]

je equal

add esi, 1

loop l1

jmped

equal:

sub edx, ecx

mov eax, edx

ed:

pop ebp

add esi, 16

ret

s2 ENDP

END main

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

1. binomial coefficients C(n, k)

**ANSWER:**

INCLUDE Irvine32.inc

.data

b byte "CO-EFFICIENT OF BIONOMIAL IS ",0

nv byte "ENTER VALUE OF N IN POSITIVE INTEGER : ",0

kv byte "ENTER VALUE OF K IN POSITIVE INTEGER (LESS THAN OR EQUALS TO N) : ",0

n DWORD ?

k DWORD ?

n\_f DWORD ?

k\_f DWORD ?

nk\_f DWORD ?

.code

main PROC

mov edx, OFFSET nv

CALL WRITESTRING

CALL READDEC

mov n, eax

mov edx, OFFSET kv

CALL WRITESTRING

CALL READDEC

mov k, eax

push n

CALL FACT

Mov n\_f, eax

push k

CALL FACT

Mov k\_f, eax

Mov eax, n

Sub eax, k

Push eax

CALL FACT

Mov nk\_f, eax

Mov eax, k\_f

Mov ebx, nk\_f

Mul ebx

Mov ecx, eax

Mov eax, n\_f

Mov edx, 0

Mov esi, ecx

Div esi

CALL CRLF

Mov edx, OFFSET b

CALL WRITESTRING

CALL WRITEDEC

CALL CRLF

exit

main ENDP

FACT PROC

Push ebp

Mov ebp, esp

Mov eax, [ebp+8]

Cmp eax, 0

ja l1

mov eax, 1

jmp l2

l1:

dec eax

push eax

CALL FACT

RETFACT:

Mov ebx, [ebp+8]

Mul ebx

l2:

pop ebp

ret 4

FACT ENDP

END main

x-------x-------x

1. 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)**;

}

**ANSWER:**

INCLUDE Irvine32.inc

.data

msg byte "K POWER OF N IS ",0

nv byte "ENTER VALUE OF N IN POSITIVE INTEGER : ",0

kv byte "ENTER VALUE OF K IN POSITIVE INTEGER : ",0

n DWORD ?

k DWORD ?

.code

main PROC

mov edx, OFFSET nv

CALL WRITESTRING

CALL READDEC

mov n, eax

mov edx, OFFSET kv

CALL WRITESTRING

CALL READDEC

mov k, eax

push n

push k

CALL POW

CALL CRLF

Mov edx, OFFSET MSG

CALL WRITESTRING

CALL WRITEDEC

CALL CRLF

exit

main ENDP

POW PROC

Push ebp

Mov ebp, esp

Mov eax, [ebp+12] ;value of n

Mov ebx, [ebp+8] ;value of k

Cmp ebx, 0

ja l1

mov eax, 1

jmp l2

l1:

dec ebx

push eax

push ebx

CALL POW

RETPOW:

Mov ecx, [ebp+12]

Mul ecx

l2:

pop ebp

ret 8

POW ENDP

END main

x-------x-------x

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

|  |  |  |
| --- | --- | --- |
| 01 | intfibonacci(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 | } |

**ANSWER:**

INCLUDE Irvine32.inc

.data

msg byte "NTH TERM OF FIBONACCI SERIES IS ",0

nv byte "ENTER NTH TERMS : ",0

n DWORD ?

.code

main PROC

mov edx, OFFSET nv

CALL WRITESTRING

CALL READDEC

mov n, eax

mov eax, 0

mov ebx, 0 ;n1

mov ecx, 1 ;n2

mov edx, 0 ;sum

push n

CALL FIBONACCI

Mov eax, edx

CALL CRLF

Mov edx, OFFSET MSG

CALL WRITESTRING

CALL WRITEDEC

CALL CRLF

exit

main ENDP

FIBONACCI PROC

Push ebp

Mov ebp, esp

Mov eax, [ebp+8] ;value of n

Cmp eax, 0

ja l

mov edx, 0

jmp l2

l:

cmp eax, 1

ja l1

mov edx, 1

jmp l2

l1:

dec eax

push eax

CALL FIBONACCI

RETFIBONACCI:

Mov esi, ebx

Add esi, ecx

Mov edx, esi

Mov ebx, ecx

Mov ecx, esi

l2:

pop ebp

ret 4

FIBONACCI ENDP

END main

x-------x-------x

(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.

**ANSWER:**

INCLUDE Irvine32.inc

.data

arrdword 9, 1, 5, 4, 2, 8, 3

lcdword 0

c\_ dword 0

tdword 0

b byte 'BEFORE',0

a byte 'AFTER', 0

.code

main PROC

mov edx, OFFSET b

mov ecx, sizeof b-1

CALL WRITESTRING

CALL CRLF

Mov esi, OFFSET arr

Mov edi, type arr

Mov ecx, lengthofarr

l0:

mov eax, [esi]

CALL WRITEDEC

Mov eax, ' '

CALL WRITECHAR

Add esi, edi

loop l0

CALL CRLF

Mov esi, OFFSET arr

Mov ecx, lengthofarr

l1:

mov lc, ecx

mov edx, c\_

mov ecx, lc

l2:

cmp ecx, 1

je no

mov esi, OFFSET arr

add esi, edx

mov eax, [esi]

mov esi, OFFSET arr

add esi, edx

add esi, edi

mov ebx, [esi]

cmp eax, ebx

jbe no

mov esi, OFFSET arr

add esi, edx

add esi, edi

mov [esi], eax

mov esi, OFFSET arr

add esi, edx

mov [esi], ebx

no:

add edx, edi

loop l2

mov ecx, lc

loop l1

CALL CRLF

Mov edx, OFFSET a

Mov ecx, sizeof a-1

CALL WRITESTRING

CALL CRLF

Mov esi, OFFSET arr

Mov ecx, lengthofarr

l3:

mov eax, [esi]

CALL WRITEDEC

Mov eax, ' '

CALL WRITECHAR

Add esi, edi

loop l3

CALL CRLF

CALL CRLF

exit

main ENDP

END main

**X----X----X**

**(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.

**ANSWER:**

INCLUDE Irvine32.inc

.data

arr dword 4, 6, 5, 4, 3, 2, 1

lc dword 0

c\_ dword 0

t dword 0

b byte 'BEFORE',0

a byte 'AFTER', 0

.code

main PROC

mov edx, OFFSET b

mov ecx, sizeof b-1

CALL WRITESTRING

CALL CRLF

Mov esi, OFFSET arr

Mov edi, type arr

Mov ecx, lengthofarr

l0:

mov eax, [esi]

CALL WRITEDEC

Mov eax, ' '

CALL WRITECHAR

Add esi, edi

loop l0

CALL CRLF

Mov esi, OFFSET arr

Mov ecx, lengthofarr

l1:

mov lc, ecx

mov edx, c\_

mov ecx, lc

l2:

cmp ecx, 1

je no

mov esi, OFFSET arr

add esi, edx

mov eax, [esi]

mov esi, OFFSET arr

add esi, edx

add esi, edi

mov ebx, [esi]

cmp eax, ebx

jbe no

mov esi, OFFSET arr

add esi, edx

add esi, edi

mov [esi], eax

mov esi, OFFSET arr

add esi, edx

mov [esi], ebx

no:

add edx, edi

loop l2

mov ecx, lc

loop l1

CALL CRLF

Mov edx, OFFSET a

Mov ecx, sizeof a-1

CALL WRITESTRING

CALL CRLF

Mov esi, OFFSET arr

Mov ecx, lengthofarr

l3:

mov eax, [esi]

CALL WRITEDEC

Mov eax, ' '

CALL WRITECHAR

Add esi, edi

loop l3

CALL CRLF

CALL CRLF

exit

main ENDP

END main

**X----X----X**

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 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 and MIPS version of this program is required.

**ANSWER:**

INCLUDE Irvine32.inc

.data

msg byte "23 \* N = ",0

nv byte "ENTER VALUE OF N : ",0

n WORD ?

.code

main PROC

mov eax, 0

mov ecx, 0

mov edx, OFFSET nv

CALL WRITESTRING

CALL READDEC

mov n, ax

shl ax, 4

add cx, ax

mov ax, n

shl ax, 2

add cx, ax

mov ax, n

shl ax, 1

add cx, ax

mov ax, n

shl ax, 0

add cx, ax

mov ax, cx

mov edx, OFFSET MSG

CALL WRITESTRING

CALL WRITEDEC

CALL CRLF

CALL CRLF

exit

main ENDP

END main

X----X----X

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: 00001FEC

|  |  |
| --- | --- |
| STACK | |
| 00001FE4h | X2 |
| 00001FE8h | X1 |
| 00001FECh | EBP |
| 00001FF0h | 11500000H |
| 00001FF4h | 5H |
| 00001FF8h | 6H |

MOV EAX, [EBP+8]

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

PUSH OFFSET X1

PUSH OFFSET X2 ; ESP: 00001FE0

POP ESI

POP EBX

ADD [ESI], EAX ; X2: 32H

ADD [EBX], EAX ; X1: 30H

MOV ESP, EBP

POP EBP

RET 8 ; EIP: 111500000H

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".

**ANSWER:**

INCLUDE Irvine32.inc

.data

source byte "This is the source string",0

des byte " ",0

.code

main PROC

mov esi, OFFSET source

mov edi, OFFSET des

mov ecx, LENGTHOF source

cld

rep MOVSB

mov edx, OFFSET des

CALL WRITESTRING

CALL CRLF

exit

main ENDP

END main

X----X----X

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.
2. 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];

voidstar\_array () {

char cell[20];

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

cell[i] = ‘\*’;

moon[i] = ‘x’;

}

}

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.
2. Using string primitives, write an assembly language program that searches 20 elements of array ArraySearchValues in 1000 un sorted elements of another array ArrayValues.
3. 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.

**ANSWER:**

INCLUDE Irvine32.inc

.data

source byte "FAST NATIONAL UNIVERSITY",0

des byte 25 dup(0)

.code

main PROC

mov eax, 0

mov esi, OFFSET source

mov edi, OFFSET des

mov ecx, LENGTHOF source

cld

rep MOVSB

mov ecx, lengthof des

mov eax, 0

mov esi, OFFSET des

l:

mov al, [esi]

Call WriteDec

Mov eax, ' '

CALL WRITECHAR

Inc esi

loop l

CALL CRLF

exit

main ENDP

END main