Nombre: José Francisco Altamirano Zevallos Clave única: 000163314 Calif:\_\_\_\_\_\_\_\_

Valor de cada pregunta: la 1 vale 3.6 pts., el resto (2, 3) valen cada una 3.2 pts.

**CONTESTAR CON TINTA**.

1. Elaboración de un programa en ensamblador que convertirá cuatro cadenas ASCII, “0”s y “1”s, a su respectiva representación binaria de un número real (punto flotante, Figura 1) y los imprima con el procedimiento de escritura de flotantes. Esto lo repetirá para cada uno de las cuatro cadenas de caracteres *cadenasreal4*, definidas con la directiva BYTE dejando su representación binaria REAL4 en el arreglo de cuatro reales *varsreal4*. En ningún momento operará los valores reales, p.e. haciendo sumas, restas o multiplicaciones.

El archivo donde usted programará en ensamblador se debe llamar “Clv*Nom***ExamFinal.asm**” donde *Clv* a su clave única y *Nom* corresponder a las iniciales de su nombre, p.e. “*967456ARBIExmFin*.asm”.

Su programa deberá contener las siguientes directivas con los valores indicados (ver *ExFinAD20ini.asm*):

.DATA

cadenasreal4 BYTE “1-10000010-11000000000000000000000”, 0

BYTE “0-10000011-01110000000000000000000”, 0

BYTE “1-10000010-11100000000000000000000”, 0

BYTE “0-10000011-10000000000000000000000”, 0

varisreal4 REAL4 0.0, 0.0, 0.0, 0.0

. . .

Recordemos que un REAL4 (32 bits) está representado bajo el siguiente formato:

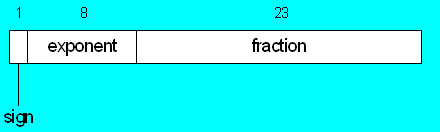


Figura 1

Elabore un procedimiento que reciba, entre otros valores, una dirección (offset) de alguna de las cadenas, y que convierta dicha cadena de caracteres (ASCIIs) en un número real para guardar en uno de los cuatro reales REAL4. Notara usted que las cadenas de caracteres están separadas por dos guiones en tres campos: el string que representa el signo SIGN, el string que representa al exponente EXPONENT y el string que representa a la mantisa FRACTION. En la figura 1 se muestra la longitud de cada campo.

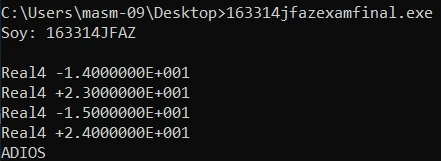
A continuación, el despliegue de la impresión de los dos primeros número reales.

|  |
| --- |
| Soy misInicialesClaveUnica  REAL4 -1.4000000E+001  REAL4 +2.3000000E+001  . . .  ADIOS. |

SIGUIENTE PÁGINA >>>>

Aunque es el proceso inverso al de este ejercicio, en ClsCB vimos el problema de convertir en ASCII el contenido binario de la representación de un número real, en el archivo “WriteBin.asm”.

Resultado:



OBSERVACIONES:

* Puede definir más de un procedimiento si así lo requiere.
* El programa deberá imprimir, al principio de la corrida, sus iniciales seguido de su clave única. El archivo deberá renombrarlo con sus iniciales y clave única, p. e. “*967456ARBIExamFinal*.asm” poniéndolo dentro de un archivo zip “*967456ARBIExamFinal*.zip” y subirlo a Comunidad, en OPC, en la sección de Trabajos y Exámenes bajo la entrada EXAMENFINAL. No incluya en el archivo “zip” el ejecutable.

SIGUIENTE PÁGINA >>>>

Nombre: José Francisco Altamirano Zevallos Clave única: 000163314 Calif:\_\_\_\_\_\_\_\_

Valor de cada pregunta: la 1 vale 3.6 pts., el resto (2, 3) valen cada una 3.2 pts.

**CONTESTAR CON TINTA**.

1. Interprete con detalle y explique gráficamente las componentes que intervienen en cada operación y los contenidos.

Exprese los resultados pedidos en hexadecimal. Indique los contenidos de los registros, paso a paso.

**A)** ¿Cuál o cuáles son los posibles resultados que pueden aparecer al final, en AX?

MOV AX, 021C2h

RCR AX, 3

Despliegue e Interpretación: Los resultados posibles son que AX tenga almacenado el valor hexadecimal: A438h o 8438h, esto se debe a que RCR significa Rotate Carry Right, la cual es una operación que desplaza todos los bits a la derecha de forma circular, incluyendo con esto el bit del carry.

Gráficamente se vería como (con x representando al bit del carry):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |  | X |
| x | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |  | 0 |
| 0 | x | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |  | 1 |
| 1 | 0 | x | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |  | 0 |

**B)** Indique los valores finales de EAX y EBX.

MOV EAX, 20212423h

MOV EBX, EAX

SHLD EAX, EBX, 4

Despliegue e Interpretación: El resultado es que EAX tendría almacenado el valor hexadecimal 2124232h y EBX 20212423h, eso se debe a que SHLD significa shift left double, y lo que hace es que desplaza todos los bits del destino hacia la izquierda un lugar, siendo que el MSB pasa al carry y el LSB es llenado por el MSB de la fuente.

Gráficamente se vería como:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EAX | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| EBX | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |

**C)** Muestre el valor final de AX.

MOV AX, 25

SHL AX, 3

Despliegue e Interpretación: El resultado es que AX tendría almacenado el valor hexadecimal C8h, esto se debe a que SHL significa shift left, y lo que hace es que mueve el valor almacenado hacia la izquierda en un bit, poniendo en el lsb un 0, es equivalente a hacer una multiplicación por 2 ^n

Gráficamente se vería como:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

1. Interprete con detalle y explique gráficamente las componentes que intervienen en cada operación y los contenidos.

Exprese los resultados pedidos en decimal. Indique los contenidos del FPU stack, paso a paso.

**A)** ¿Cuál o cuáles son los posibles resultados que pueden aparecer al final?, tanto en las localidades de memoria (DATA) como en el FPU stack?

.DATA

uno REAL8 -2000.0

dos REAL8 300.0

tres REAL8 -4.0

.CODE

fldz

fld uno

fabs

fdiv dos

fstp tres

call ShowFPUStack

Despliegue e Interpretación: El FPUStack paso a paso se vería como:

|  |  |  |
| --- | --- | --- |
| fldz | ST(0) | 0 |
|  |  |  |
| fld uno | ST(0) | -2000 |
|  | ST(1) | 0 |
|  |  |  |
| fabs | ST(0) | 2000 |
|  | ST(1) | 0 |
|  |  |  |
| fdiv | ST(0) | 6.6666666 |
|  | ST(1) | 0 |
|  |  |  |
| fstp tres | ST(0) | 0 |
|  |  |  |
| showFPUStack | ST(0) | 0 |
|  |  |  |

Las localidades de memoria uno y dos no se verían afectadas y en tres, estaría almacenado el valor 6.666666

**B)** ¿Cuál o cuáles son los posibles resultados que pueden aparecer al final?, tanto en las localidades de memoria (DATA) como en el FPU stack?

.DATA

cuatro REAL8 3.0

cinco REAL8 4.0

seis REAL8 5.0

siete REAL8 -6.0

.CODE

fld cuatro

fchs

fld cinco

fmul seis

fadd

fst siete

call ShowFPUStack

Despliegue e Interpretación: El FPUStack paso a paso se vería como:

|  |  |  |
| --- | --- | --- |
| fld cuatro | ST(0) | 3 |
|  |  |  |
| fchs | ST(0) | -3 |
|  |  |  |
| fld cinco | ST(0) | 4 |
|  | ST(1) | -3 |
|  |  |  |
| fmul seis | ST(0) | 20 |
|  | ST(1) | -3 |
|  |  |  |
| fadd | ST(0) | 17 |
|  |  |  |
| fst siete | ST(0) | 17 |
|  |  |  |
| showFPUStack | ST(0) | 17 |

Las localidades de memoria cuatro, cinco y seis no se verían afectadas y en siete estaría guardado el valor 17