**TALLER (PRIMER PARCIAL)**

**ARQUITECTURA DE COMPUTADORES**

**2018**

**Jornada Especial.**

1. ¿Que es una arquitectura de computadores?

R/ La arquitectura de computadoras se trata del diseño y la estructura operacional fundamental del sistema que componen a una computadora. Es la descripción de los requerimientos de diseño para cambiar partes de una computadora, haciendo énfasis en la forma en que la unidad central de proceso trabaja internamente e ingresa a las direcciones de memoria.

1. Nombre las generaciones de los computadores y sus características más relevantes.

R/ **Primera generación:**

En esta generación se usaban tubos al vacío para procesar información. Al igual quetarjetas perforadas para entrar los datos y los programas.También se usaban cilindros magnéticos para almacenar información e instrucciones internas y eran extremadamente grandes, también utilizaban enormes cantidades de electricidad y generaban calor en exceso, además eran sumamente lentas.

Se implementó el sistema binario en la representación de los datos.

**Segunda generación:**

Aquí muchas de las computadoras ya se programaban con tarjetas perforadas y otras utilizaban sistemas cableados. Ya se usaban transistores en el procesamiento de información, ya que estos eran más rápidos y pequeños que los tubos al vacío.

Usaban pequeños anillos magnéticos para almacenar información e instrucciones.Los programas de computadoras que fueron desarrollados durante la primera generación se mejorarony se desarrollaron nuevos lenguajes de programación como COBOL y FORTRAN, los cuales eran comercialmente más accesibles.

**Tercera generación:**

Emerge la industria del "software". En esta generación se desarrollaron los circuitos integrados para procesar información al igual que los "chips" para almacenar y procesar la información. En este momento las computadoras ya podían llevar a cabo ambas tareas de procesamiento o análisis matemáticos.

**Cuarta generación:**

En este tiempo se desarrolló el microprocesador y se colocaron más circuitos dentro de un "chip".

Se implementó la LSI (Large Scale Integration circuit) y laVLSI (Very Large Scale Integration circuit).

**Quinta generación:**

Surge la competencia internacional por el dominio del mercado de la computación, al notarse el acelerado crecimiento de la microelectrónica, la industria se dio a la tarea de poner a la altura el desarrollo del software y los sistemas con que se manejan las computadoras.

Se desarrollan las computadoras personales o PC y se desarrollan las supercomputadoras.

1. Segun Flynn ¿Cuál es la clasificación de las arquitecturas?

|  |  |  |
| --- | --- | --- |
|  | Una instrucción | Múltiples instrucciones |
| Un dato | SISD | MISD |
| Múltiples datos | SIMD | MIMD |

**Single Instruction, Single Data**: Es un único procesador que interpreta una única secuencia de instrucciones para operar con los datos almacenados en una única memoria. La CPU procesa únicamente una instrucción por cada ciclo de reloj

**Single Instruction, Múltiple Data:** Una secuencia de instrucciones y múltiples secuencias de datos.

Todas las unidades ejecutan la misma instrucción entonces cada unidad procesa un dato distinto y todas las unidades operan simultáneamente

**Multiple Instruction, Single Data:** Múltiples secuencias de instrucción y una secuencia de datos

Cada unidad ejecuta una instrucción distintay cada unidad procesa el mismo dato**,** esta aplicación es muy limitada en la vida real

**Multiple Instruction, Multiple Data:** Un conjunto de procesadores que ejecutan simultáneamente secuencias de instrucción diferentes con conjuntos de datos diferentes.

Cada unidad ejecuta una instrucción distinta y cada unidad procesa un dato distinto entonces todas las unidades operan simultáneamente

1. Nombre las clases de aplicaciones de cómputo.

R/ Las aplicaciones de la computación son

Aplicaciones de Sistema de control y automatización industrial

Aplicaciones ofimáticas

Software educativo

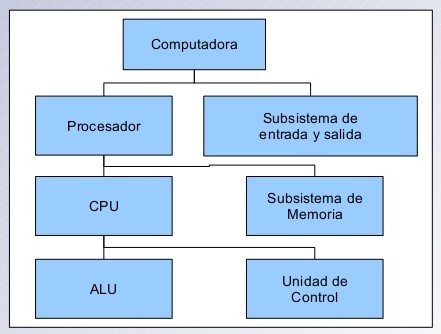
Software médico

Software de Cálculo Numérico

Software de Diseño Asistido

Software de Control Numérico

1. Muestre la clasificación de la jerarquía de un equipo de cómputo.



1. ¿Qué es un compilador?

R/ El compilador es el encargado de traducir los programas escritos en un lenguaje de programación, generando un programa que la máquina será capaz de interpretar, lenguaje de máquina, pero también puede ser un código intermedio y esto es lo que conocemos como compilación. De esta manera un programador puede diseñar un programa en un lenguaje mucho más cercano a cómo piensa un ser humano, para luego compilarlo a un programa más manejable por una computadora.

1. ¿ Defina qué es una instrucción?

R/ Se denomina instrucción en informática al conjunto de datos insertados en una secuencia estructurada o específica que el procesador interpreta y ejecuta.

Estas instrucciones del computador son las que determinan el funcionamiento de la CPU que las ejecuta. La CPU puede realizar una diversidad de funciones, que son el reflejo de la variedad de las instrucciones definidas para dicha CPU. El programador tiene un repertorio de instrucciones como medio para controlar la CPU.

1. ¿Cuales son los principios básicos de diseño de hardware de una arquitectura de cómputo, escriba una definición de cada uno?

* La simplicidad favorece la regularidad
* Entre más pequeño es más rápido
* Hacer el caso común más rápido
* Buenos diseños demandan grandes compromisos

1. ¿Qué es SPARCV8?

SPARC (Scalable Processor Architecture) es una arquitectura de microprocesador de 32 y 64 bits de Sun Microsystems que se basa en la computación reducida de conjuntos de instrucciones (RISC). SPARC se ha convertido en una arquitectura ampliamente utilizada para hardware utilizado con sistemas operativos basados ​​en UNIX, incluidos los propios sistemas Solaris de Sun. Sun ha hecho de SPARC una arquitectura abierta que está disponible para licencias a los fabricantes de microprocesadores. En su marca más reciente, UltraSPARC, los microprocesadores se pueden construir para placas de PC (ya sea mediante Interconexión de Componentes Periféricos o ATX) así como también para el mercado de estación de trabajo original de SPARC. Como evidencia de la escalabilidad de SPARC, Sun dice que su UltraSPARC III se diseñará para permitir que hasta 1.000 procesadores trabajen juntos.

SPARC es una arquitectura de conjunto de instrucciones de CPU (ISA), derivada de una reducción linaje de conjunto de instrucciones (RISC). Como arquitectura, SPARC permite un espectro de implementaciones de chip y sistema a una variedad de precio / rendimiento puntos para una variedad de aplicaciones, incluida la programación científica / de ingeniería, en tiempo real y comercial.

1. ¿Cuáles son las categorías de instrucciones de la arquitectura SPARCV8?

Las instrucciones están divididas en seis categorías básicas las cuales son:

* Load/store
* Arithmetic/logical/shift
* Control transfer
* Read/write control register
* Floating-point opérate
* Coprocessor operate

1. ¿Qué tipos de registros se encuentran en SPARC V8?

Los procesadores en SPARC tienen los siguientes tipos de registros: registros de datos y registros de control / estado.

**Registro del estado del procesador**

* Ventana máscara inválida
* Registro de la base de trampas
* Registro de multiplicación / división
* Contadores de programa
* Registros de estado auxiliar dependientes de la implementación
* IU Deferred-Trap Queue dependiente de la implementación

**Los registros de control / estado de FPU incluyen:**

* Registro Estatal de Punto Flotante (FSR)
* Cola de trampa diferida de punto flotante dependiente de la implementación (FQ)

**Los registros de control / estado del coprocesador (CP), si están presentes, pueden incluir:**

* Registro estatal de coprocesador dependiente de la implementación (CSR)
* Coprocesador dependiente de la implementación Deferred-Trap Queue (CQ)

1. ¿Cuál es el número mínimo y máximo de registros que se puede implementar en la arquitectura SPARCV8?

Los registros r de 32 bits. Están divididos en 8 registros globales, más un número de 16 registros dependientes de la implementación

La cantidad de ventanas o conjuntos de registros, NWINDOWS, varía de 2 a 32, dependiendo de la implementación. El número mínimo de registros r es 40 (2 conjuntos), y el máximo el número es 520 (32 juegos).

1. ¿Cuáles son las instrucciones de acceso a memoria de SPARCV8? de un ejemplo de cada uno.

R/ las instrucciones para acceder a la memoria son LOAD y STORE y son las únicas por medio de las cuales se puede acceder a la memoria

Ejemplos

**Load**

int d [20];

LD [%L4 +(20\*4)], L5

**Store**

c [5] = i + 2;

ST %L6, [%L2+(5\*4)]

1. Represente los siguientes números en complemento a 2.

a.5

b.12890

c.56900

d.11

e.140

1. Explique las instrucciones aritmético lógicas y su sintaxis en lenguaje ensamblador.

Ejecutan operaciones aritméticas, lógicas y de desplazamiento de bits, estas instrucciones calculan el resultado si es una función de dos operandos y guardan el resultado en un registro

1. Explique cada uno de los campos de los 3 formatos de la arquitectura SPARC V8.

Formato 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| op | rd | Op3 | Rs1 | i | Unused (zero) | Rs2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| op | rd | Op3 | Rs1 | i | Simm13 |

Formato 2

|  |  |  |  |
| --- | --- | --- | --- |
| op | rd | Op2 | Imm22 |

Op= tipo de instrucción

Rd= registro destino

Op3= tipo de operación

Re1= registro fuente 1

I= si incluye o no un inmediato o entero

Inused zero= se llena con seros en caso de no existir un inmediato

Simm13 = inmediato

1. ¿Qué diferencia hay entre el campo **op, op2 y op3**?

El **op** se refiere a el tipo de instrucció mientras que el **op2**  y el **op3** se refiere a el tipo de operacion

1. ¿Qué es **PSR** ?, explique cada uno de sus campos.
2. ¿ Qué es **ICC** y **CWP**?
3. ¿Qué es una instrucción sintética, de dos ejemplos?
4. ¿Qué significa el campo **a** para una instrucción BRANCH?.

El campo **a** en un BRANCH es el que nos indica si la instrucción se ejecuta antes o después de la evaluación de ciclo.

1. ¿Para qué la instrucción **CALL** utiliza el registro %O7?.
2. Convertir el siguiente programa en lenguaje de máquina a lenguaje ensamblador y luego a lenguaje de alto nivel:

10100000000100000010000000000101

10100010000100000011111111111010

10010000000001000100000000010000

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10001 | 000010 | 00000 | 1 | 1111111111010 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10000 | 000010 | 00000 | 1 | 0000000000101 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 000000 | 10001 | 0 | 00000000 | 10000 |

LENGUAJE ENSAMBLADOR

OR %G0, 5, %L0

OR %G0, -6, %L1

ADD %L1,%L0,0O

LENGUAJE ALTO NIVEL

INT (){

INT A = 5;

INT B = -6;

C = (-6+5);

RETURN C;

}

COMPLEMENTO A 2 PARA -6

1111111111010

0000000000101

1

00000000000110

24.Solucione el siguiente programas en lenguaje ensamblador, lenguaje de máquina y hexadecimal, además coloque su dirección de memoria.

int main(){

int i = 5;

int b = -4;

int c[100];

int d[20];

c[5] = i + 2;

d[4] = b + 3;

return c[5] + d[4] -i

}

0000 MOV 5, %L0

0004 MOV -4, %L1

0008 LD [%L2 +(100\*4)],L3

000C LD [%L4 +(20\*4)],L5

0010 ADD %L0,2,%L6

0014 ST %L6,[%L2+(5\*4)]

0018 ADD %L1,3,%L7

001C ST %L1,[%L4+(4\*4)]

0020 LD [%L2 +(5\*4)],L6

0024 LD [%L4 +(4\*4)],L7

0028 ADD L6,L7,L6

002C SUB L6,L0,O0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10000 | 000010 | 00000 | 1 | 0000000000101 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10001 | 000010 | 00000 | 1 | 1111111111100 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | 10011 | 000000 | 10010 | 1 | 0000001100100 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | 10101 | 000000 | 10100 | 1 | 0000000010100 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10110 | 000000 | 10000 | 1 | 0000000000010 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | 10110 | 000100 | 10100 | 1 | 0000000000101 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10111 | 000000 | 10001 | 1 | 0000000000011 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | 10111 | 000100 | 10100 | 1 | 0000000000100 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | 10110 | 000000 | 10010 | 1 | 0000000000101 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | 10111 | 000000 | 10100 | 1 | 0000000000100 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 10110 | 000000 | 10110 | 0 | 00000000 | 10111 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 000100 | 10110 | 0 | 00000000 | 10000 |

Complemento a dos para el -6

0000000000100

1111111111011

1

1111111111100

1. Convierta el siguiente código a lenguaje ensamblador, máquina **SPARC V8** y hexadecimal. a.

int main(){

int a = 8;

int b = -14800;

int c = 33;

if((a+b)<=b\*16){

c=a+(b\*2);

}

else{

return b-78;

}

return a+c;

}

b.

int main(){

int a = 8;

int b = -10;

if(a!=b){

return c/16;

}

else{

return b\*32;

}

}

c.

int main(){

int a = -21180;

}

R/

A)

int main(){

int a = 8;

int b = -14800;

int c = 33;

if((a+b)<=b\*16){

c=a+(b\*2);

}

else{

return b-78;

}

return a+c;

}

LENGUAJE ENSAMBLADOR

0000 MOV 8,%L0

0004 STEHI -29,%L1

0008 OR %L1,48,L1

000C MOV 33,%L3

0010 ADD %L0,%L1,%L4

0014 SLL %L1,4,%L5

0018 SUBcc %L4,%L5,G0 = CMP %L4,L5

001C BGE a SUMA

0020 SUB %L1,78,%L1

0024 BA a EXIT

SUMA 0028 SLL %L1,1,L6

002C ADD %L0,%L6,%L3

EXIT 0030 ADD %L0,%L3,O0

LENGUAJE MAQUINA

FRTO3

OP RD OP3 RS1 I SIMM13/RS2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10000 | 000010 | 00000 | 1 | 0000000001000 |

0XA0102008

FRTO2

OP RD OP2 OPERATION

|  |  |  |  |
| --- | --- | --- | --- |
| 00 | 10001 | 100 | 1111111111111111100011 |

0X2300001D

FRTO 3

OP RD OP3 RS1 I SIMM13/RS2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10001 | 000010 | 10001 | 1 | 0000000110000 |

0XA2146030

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10011 | 000010 | 00000 | 1 | 0000000100001 |

0XA6102021

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 10101 | 100101 | 10001 | 1 | 00000000 | 00100 |

0XAB2C6004

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 010100 | 10100 | 0 | 00000000 | 10101 |

0X80A50015

FRTO 2

OP a COND OP2 DISP22

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1011 | 010 | 0000000000000000000010 |

0X36800002

FRTO 3

OP RD OP3 RS1 I SIMM13/RS2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10001 | 000100 | 10001 | 1 | 0000001001110 |

0XA224604E

FRTO 2

OP a COND OP2 DISP22

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1000 | 010 | 0000000000000000000010 |

0X30800002

FRT 3

OP RD OP3 RS1 I SIMM13/RS2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 10110 | 100101 | 10001 | 1 | 00000000 | 00001 |

SUMA 0XAB2C6001

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 10011 | 000000 | 10000 | 0 | 00000000 | 10110 |

0XA6040016

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 000000 | 10000 | 0 | 00000000 | 10011 |

EXIT 0X90040013

COMPLEMENTO A DOS PARA 14800

11100111010000

00011000101111

1

00011|000110000

11100

1

11101 = 29

B)

int main(){

int a = 8;

int b = -10;

int c = 64;

if(a!=b){

return c/16;

}

else{

return b\*32;

}

}

LENGUAJE ENSAMBLADOR:

0000 MOV 8,%L0

0004 MOV -10,%L1

0008 MOV 64,%L2

000C SUBCC %L0,%L1,G0

0010 BNE a DIV

0014 SLL %L1,5,O0

0018 BA a EXIT

DIV 001C SRL %L2,4,O0

EXIT 0020 NOP

LENGUAJE DE MAQUINA:

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10000 | 000010 | 00000 | 1 | 0000000001000 |

OP RD OP3 RS1 I SIMM13/RS2

0XA0102008

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10001 | 000010 | 00000 | 1 | 1111111110110 |

0XA2103FF6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10010 | 000010 | 00000 | 1 | 0000001000000 |

0XA4102040

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 010100 | 10000 | 0 | 00000000 | 10001 |

0X80A40011

FRTO 2

OP a COND OP2 DISP22

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1001 | 010 | 0000000000000000000010 |

0X32800002

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 100101 | 10000 | 1 | 0000000000101 |

OP RD OP3 RS1 I SIMM13/RS2

0X912C2005

FRTO 2

OP a COND OP2 DISP22

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1000 | 010 | 0000000000000000000001 |

0X30800001

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 100110 | 10010 | 1 | 0000000000100 |

OP RD OP3 RS1 I SIMM13/RS2

DIV 0X9134A004

FRTO2

OP RD OP2 OPERATION

|  |  |  |  |
| --- | --- | --- | --- |
| 00 | 00000 | 100 | 0000000000000000000000 |

EXIT 0X01000000

COMPLEMENTO A 2 PARA -10

01010

10101

1

10110

C)

int main(){

int a = -21180;

}

LENGUAJE ENSAMBLADOR

0000 STEHI -21,%L0

0004 OR %L0,324,%L0

LENGUAJE DE MAQUINA

|  |  |  |  |
| --- | --- | --- | --- |
| 00 | 10000 | 100 | 1111111111111111101011 |

0X213FFFEB

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10000 | 000010 | 10000 | 1 | 0000101000100 |

0XA0142144

COMPLEMENTO A DOS PARA -21180

101001010111100

010110101000011

1

01011|0101000100

10100

1

10101 = -21

1. Convierta el siguiente código a lenguaje ensamblador, máquina **SPARC V8** y hexadecimal.

int test(int x, int y, int w){

int z;

z = x - y + w\*4;

return z + 2;

}

int main(){

int a = 4, b = 2, c = -15600;

int x = test(a,b,c);

return x + 45;

}

LENGUAJE ENSAMBLADOR

TEST 0000 SUB %I0,%I1,%I0

0004 SLL %I3,2,%I3

0008 ADD %I3,%I0,%O0

000C JMPL %07+8,%G0

0010 ADD %O0,2,%O0

MAIN 0014 MOV 4,%I0

0018 MOV 2,%I1

001C SETHI -31,%I3

0020 OR %I3,272,%I3

0024 CALL TEST

0028 MOV 0,%L0

002C ADD %L0,45,%01

LENGUAJE MAQUINA

TEST:

OP RD OP3 RS1 I SIMM13/RS2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 11000 | 000100 | 11000 | 0 | 00000000 | 11001 |

0XB0260019

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 11011 | 100101 | 11011 | 1 | 00000000 | 00010 |

0XB72EE002

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 000000 | 11011 | 0 | 00000000 | 11000 |

0X9006C018

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 111000 | 01111 | 1 | 0000000001000 |

0X81C3E008

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 000000 | 01000 | 1 | 0000000000010 |

0X90022002

MAIN:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 11000 | 000010 | 00000 | 1 | 0000000000100 |

OP RD OP3 RS1 I SIMM13/RS2

0XB0102004

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 11001 | 000010 | 00000 | 1 | 0000000000010 |

0XB2102002

FRTO 2

OP RD OP2 OPERATION

0X373FFFE1

|  |  |  |  |
| --- | --- | --- | --- |
| 00 | 11011 | 100 | 1111111111111111100001 |

FRTO 3

OP RD OP3 RS1 I SIMM13/RS2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 11011 | 000010 | 11011 | 1 | 0000100010000 |

0XB616E110

FRTO 1

OP DISP 30

|  |  |
| --- | --- |
| 01 | 111111111111111111111111110111 |

0X7FFFFFF7

FRTO3

OP RD OP3 RS1 I SIMM13/RS2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10000 | 000010 | 10000 | 1 | 0000000000000 |

0XA0142000

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 000000 | 10000 | 1 | 0000000101101 |

0X9004202D

COMPLEMENTO A DOS PARA LOS DISP DEL CALL -9

1001

0110

1

0111

COMPLEMETNO A 2 PARA -15600

11110011110000

00001100001111

1

00001|100010000

11110

1

11111 =-31

1. Implemente la función **Pot** en lenguaje de alto nivel,lenguaje ensamblador **SPARC V8** y lenguaje de máquina SPARC V8 que realice la potencia de dos números enteros sin signo realizando llamados a la función multiplicacion hecha en clase.

int mul(int c, int d){

int z=0;

for(int cont=1;cont<=d;cont+=1){

z=z+c;

}

return z;

}

int pot(int a,int b){

int c = a;

if(b==0){

return a=1;

}

else{

for(int i=1;i<b;i++){

a=mul(a,c);

}

return a;

}

}

Int main(){

Int a=4, b=8;

Int x= pot(a,b);

Return x;

}

LENGUAJE ENSAMBLADOR:

MUL 0000 MOV 0,%O0

0004 MOV 1,%L0

FOR 0008 SUBCC %I0,%L0,G0 == CMP %I0,%L0

000C BGE a SALIR

0010 ADD %O0,%L2,%O0

0014 BA a FOR

SALIR 0018 JMPL %O7+8,%G0

POT 001C MOV %I1,%L1

0020 MOV %L2,1

0024 SUBCC %I2,%L3,G0

0028 BNE a ELSE

002C MOV 1,%I1

ELSE 0030 SUBCC %L2,I2,GO

0034 BGE SALIRE

0038 CALL MUL

003C NOP

0040 BA a ELSE

SALIRE 0044 JMPL %O6+8,G0

MAIN 0048 MOV 4,%I1

004C MOV 8,%I2

0050 CALL POT

0054 MOV 0,O1

LENGUAJE MAQUINA

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 000010 | 00000 | 1 | 0000000000000 |

0X90102000

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10000 | 000010 | 00000 | 1 | 0000000000001 |

0XA0102001

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 010100 | 11000 | 0 | 00000000 | 10000 |

0X80A60010

FRTO2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1011 | 010 | 0000000000000000000010 |

0X36800002

FRTO 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 000000 | 01000 | 0 | 00000000 | 10010 |

0X90020012

FRTO 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1000 | 010 | 1111111111111111111101 |

0X30BFFFFD

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 111000 | 01111 | 1 | 0000000001000 |

0X81C3E008--

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 10001 | 000010 | 11001 | 0 | 00000000 | 00000 |

0XA2164000

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10010 | 000010 | 10010 | 1 | 0000000000000 |

0XA414C000

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 010100 | 11010 | 0 | 00000000 | 10011 |

0X80A68013

FRTO2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1011 | 010 | 0000000000000000000001 |

0X36800001

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 11001 | 000010 | 00000 | 1 | 0000000000001 |

0XB2102001

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 010100 | 10010 | 0 | 00000000 | 11010 |

0X80A4801A

FRTO 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1011 | 010 | 0000000000000000000001 |

0X36800001

FRTO 1

|  |  |
| --- | --- |
| 01 | 111111111111111111111111110010 |

0X7FFFFFF2

FRTO2

|  |  |  |  |
| --- | --- | --- | --- |
| 00 | 00000 | 100 | 0000000000000000000000 |

0X01000000

FRTO2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1000 | 010 | 1111111111111111111100 |

0X30BFFFFC

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 111000 | 01110 | 1 | 0000000001000 |

0X81C3A008

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 11001 | 000010 | 00000 | 1 | 0000000000100 |

0XB2102004

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 11010 | 000010 | 00000 | 1 | 0000000001000 |

0XB4102008

FRTO 1

|  |  |
| --- | --- |
| 01 | 111111111111111111111111110101 |

X07FFFFFF6

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 01001 | 000010 | 00000 | 1 | 0000000000000 |

0X92102000

1. Implemente una función **Fact** en lenguaje de alto nivel, lenguaje ensamblador **SPARC V8** y lenguaje de máquina SPARC V8 que calcule el factorial de un número entero sin signo.

int mul(int c, int d){

int z=0;

for(int cont=1;cont<=d;cont+=1){

z=z+c;

}

return z;

}

int fact (int a){

int b,fac=1;

for (b=1 ; b<=a ; b++)

{

fac=mul(b,fac);

}

return fac;

}

Int main(){

Int a=7

Int x= fact(a);

Return x;

}

LENGUAJE ENSAMBLADOR:

MUL 0000 MOV 0,%O0

0004 MOV 1,%L0

FOR 0008 SUBCC %I0,%L0,G0 == CMP %I0,%L0

000C BGE a SALIR

0010 ADD %O0,%L2,%O0

0014 BA a FOR

SALIR 0018 JMPL %O7+8,%G0

FACT 001C MOV %L1,1

0020 MOV %L2,1

FOR1 0024 SUBCC %L1,%I1,G0’

000C BGE a SALIR1

0010 CALL MUL

0000 NOP

0014 BA a FOR1

SALIR1 0018 JMPL %O6+8,%G0

MAIN 0000 MOV 7,%I1

0000 CALL FACT

0000 MOV 0,%01

LENGUAJE MAQUINA

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 000010 | 00000 | 1 | 0000000000000 |

0X90102000

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10000 | 000010 | 00000 | 1 | 0000000000001 |

0XA0102001

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 010100 | 11000 | 0 | 00000000 | 10000 |

0X80A60010

FRTO2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1011 | 010 | 0000000000000000000010 |

0X36800002

FRTO 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 01000 | 000000 | 01000 | 0 | 00000000 | 10010 |

0X90020012

FRTO 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1000 | 010 | 1111111111111111111101 |

0X30BFFFFD

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 111000 | 01111 | 1 | 0000000001000 |

0X81C3E008

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10001 | 000010 | 00000 | 1 | 0000000000001 |

0XC2102001

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10010 | 000010 | 00000 | 1 | 0000000000001 |

0XC4102001

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 010100 | 10001 | 0 | 00000000 | 11001 |

0X80A44019

FRTO2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1011 | 010 | 0000000000000000000011 |

0X36800003

FRTO 1

|  |  |
| --- | --- |
| 01 | 111111111111111111111111110101 |

0X7FFFFFF6

FRTO2

|  |  |  |  |
| --- | --- | --- | --- |
| 00 | 00000 | 100 | 0000000000000000000000 |

0X01000000

FRTO2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 00 | 1 | 1000 | 010 | 1111111111111111111001 |

0X30BFFFF9

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 00000 | 111000 | 01110 | 1 | 0000000001000 |

0X81C3A008

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 11001 | 000010 | 00000 | 1 | 0000000000111 |

0XB2102007

FRTO 1

|  |  |
| --- | --- |
| 01 | 111111111111111111111111110111 |

0X7FFFFFF7

FRTO 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 01001 | 000010 | 00000 | 1 | 0000000000000 |

0X92102000