**1.- Completa las siguientes tablas:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BINARIO** | (110010101110)₂ | (001110)₂ | (1111011)₂ | **11111111** |
| **DECIMAL** | (3246)₁₀ | (14)₁₀ | **123** | (255)₁₀ |
| **OCTAL** | (6256)₈ | **16** | (173)₈ | (377)₈ |
| **HEXADECIMAL** | **CAE** | E | (7B)₁₆ | (FF)₁₆ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BINARIO** | **10101001** | (1111011011)₂ | (1010111101)₂ | (111111101010)₂ |
| **DECIMAL** | (169)₁₀ | **987** | (449)₁₀ | (4074)₁₀ |
| **OCTAL** | (6256)₈ | (6256)₈ | **701** | (6256)₈ |
| **HEXADECIMAL** | (A9)₁₆ | (3DB)₁₆ | (1C1)₁₆ | FEA |

**¿Cómo se resuelve la tabla anterior?... Lo primero que debemos tener en cuenta son estos datos, pues son fundamentales:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DECIMAL** | **BINARIO** | **HEXADECIMAL** |  | **BINARIO** | **OCTAL** |
| 0 | 0000 | 0 | 000 | 0 |
| 1 | 0001 | 1 | 001 | 1 |
| 2 | 0010 | 2 | 010 | 2 |
| 3 | 0011 | 3 | 011 | 3 |
| 4 | 0100 | 4 | 100 | 4 |
| 5 | 0101 | 5 | 101 | 5 |
| 6 | 0110 | 6 | 110 | 6 |
| 7 | 0111 | 7 | 111 | 7 |
| 8 | 1000 | 8 |  |  |
| 9 | 1001 | 9 |  |  |
| 10 | 1010 | A |  |  |
| 11 | 1011 | B |  |  |
| 12 | 1100 | C |  |  |
| 13 | 1101 | D |  |  |
| 14 | 1110 | E |  |  |
| 15 | 1111 | F |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **2⁷** | **2⁶** | **2⁵** | **2⁴** | **2³** | **2²** | **2¹** | **2°** |
| **128** | **64** | **32** | **16** | **8** | **4** | **2** | **1** |
| **1 ó 0** | **1 ó 0** | **1 ó 0** | **1 ó 0** | **1 ó 0** | **1 ó 0** | **1 ó 0** | **1 ó 0** |

**Si 1 = Activado / Encendido**

**Si 0 = Desactivado / Apagado**

**HEXADECIMAL (CAE)₁₆ 🡪 OCTAL**

Lo primero que vamos a hacer para facilitarnos la tarea (estamos empezando con estas conversiones), va a ser convertir HEXADECIMAL a BINARIO y luego de BINARIO a OCTAL:

**1º)**

C = 1100 ; A = 1010; E = 1110

**(110010101110)₂** 🡪 Convertido a **BINARIO** (base 2)

**2º)** Como es **OCTAL** hacemos grupos de 3 bits:

110|010|101|110

**3º)** Por último, calculamos valor por GRUPO de 3 bits:

110|010|101|110 🡪 **(6256)₈** 🡪 Base 8 = **OCTAL**

**6 2 5 6**

**HEXADECIMAL (CAE)₁₆ 🡪 DECIMAL**

**1º)** Empezamos trabajando con base 16 porque HEXADECIMAL usa base 16, por tanto:

(CAE)₁₆ 🡪 C = 12; A = 10; E = 14

**2º)** Ahora posicionamos de derecha a izquierda

(C A E)₁₆

2 1 0

**3º)** Calculamos valores en base 16 exponenciando su POSICION correspondiente de Arriba Abajo:

C 🡪 12 x 16² = 12 x 256 🡪 C = 3072

A 🡪 10 x 16¹ = 10 x 16 🡪 A = 160 C + A +E = **(3246)₁₀**

E 🡪 14 x 16° = 14 x 1 🡪 E = 14

**Nota: Cualquier número elevado a 0 es igual a 1 🡪 16° = 1**

**OCTAL (16)₈ 🡪 DECIMAL**

(1 6)₈ 🡪 1 X 8¹ = 8 (8 + 6) = **(14)₁₀**

1 0 6 x 8° = 6

**OCTAL (16)₈ 🡪 BINARIO**

Pasamos **(16)₈** a **BINARIO** (base 2) en GRUPOS de 3 bits cada uno (porque estamos usando OCTAL, es decir, base 8), y los **OCTALES** se dividen en grupos de 3 bits

De izquierda a derecha:

1 🡪 001

16 🡪 110 (001110)₂

**OCTAL (16)₈ 🡪 HEXADECIMAL**

**HEXADECIMAL** funciona de manera muy parecida a **OCTAL** pero con base 16 (OCTAL es base 8), por tanto, los grupos son de 4 bits cada uno.

Las letras llegan hasta 15, en este caso, es superiora a 15 (16), por tanto, no podemos usar letras directamente:

(16)₈ = (001|110)₂ 🡪 (0000|1110)₂ 🡪 **E**

14

**DECIMAL (123)₁₀ 🡪 BINARIO**

(123)₁₀ 🡪 ( 1 1 1 1 0 1 1 )₂

64 32 16 8 4 2 1

**DECIMAL (123)₁₀ 🡪 OCTAL**

|  |  |
| --- | --- |
| **PRODUCTO** | **RESTO** |
| 15 | 3 |
| 1 | 7 |
| 0 | 1 |

(123)₁₀ 🡪 123/8

15/8

1/8

(123)₁₀ 🡪 (173)₈

**DECIMAL (123)₁₀ 🡪 HEXADECIMAL**

|  |  |
| --- | --- |
| **PRODUCTO** | **RESTO** |
| 7 | 11 |
| 0 | 7 |
|  |  |

(123)₁₀ 🡪 123/16

7/16

(123)₁₀ = **(7B)₁₆**

11

**BINARIO (11111111)₂ 🡪 DECIMAL**

2⁷ 2⁶ 2⁵ 2⁴ 2³ 2² 2¹ 2°

(1 1 1 1 1 1 1 1)₂ 🡪 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = **(255)₁₀**

**BINARIO (11111111)₂ 🡪 OCTAL**

(011|111|111)₂ 🡪 **(377)₈**

3 7 7

**BINARIO (11111111)₂ 🡪 HEXADECIMAL**

(1111|1111)₂ 🡪 **(FF)₁₆**

15 15

**BINARIO (10101001)₂ 🡪 DECIMAL**

(1 0 1 0 1 0 0 1)₂ 🡪**(169)₁₀**

128 64 32 16 8 4 2 1

**BINARIO (10101001)₂ 🡪 OCTAL**

(010|101|001) 🡪 **(251)₈**

2 5 1

**BINARIO (10101001)₂ 🡪 HEXADECIMAL**

(1010|1001) 🡪 **(A9)₁₆**

10 9

A

**DECIMAL (987)₁₀ 🡪 BINARIO**

(987)₁₀ 🡪 2ꝰ + 2⁸ + 2⁷ + 2⁶ + 2⁵ + 2⁴ + 2ᶟ + 2² + 2¹ + 2° = **(987)₁₀**

1 1 1 1 0 1 1 0 1 1 = (1 1 1 1 0 1 1 0 1 1)₂ = **(987)₁₀**

**DECIMAL 🡪 OCTAL**

|  |  |
| --- | --- |
| PRODUCTO | RESTO |
| 123 | 3 |
| 15 | 3 |
| 1 | 7 |
| 0 | 1 |

(987)₁₀ 987/8

123/8

15/8

1/8

(987)₁₀ = **(1733)₈**

**DECIMAL (987)₁₀ 🡪 HEXADECIMAL**

|  |  |
| --- | --- |
| PRODUCTO | RESTO |
| 61 | 11 |
| 3 | 13 |
| 0 | 3 |
|  |  |

(987)₁₀ 987/16

123/16

15/16

(987)₁₀ = **(3DB)₁₆**

**OCTAL (701)₈ 🡪 BINARIO**

(701)₈ 🡪 2ꝰ + 2⁸ + 2⁷ + 2⁶ + 2⁵ + 2⁴ + 2ᶟ + 2² + 2¹ + 2°

**(1 0 1 0 1 1 1 1 0 1)₂**

**OCTAL (701)₈ 🡪 DECIMAL**

7 x 8² = 448

(7 0 1)₈ 0 x 8¹ = 0 448 + 0 + 1 = **(449)₁₀**

2 1 0 1 x 8° = 1

**OCTAL (701)₈ 🡪 HEXADECIMAL**

(701)₈ 🡪 0001|1100|0001 🡪 **(1C1)₁₆**

1 12 1

**HEXADECIMAL (FEA)₁₆ 🡪 OCTAL**

(FEA)₁₆ 🡪 111|111|101|010 🡪 **(7752)₈**

7 7 5 2

**HEXADECIMAL (FEA)₁₆ 🡪 DECIMAL**

F = 15 15 x 16² = 3840

(F E A)₁₆ E = 14 14 x 16¹ = 224 3840 + 224 + 10 = **(4074)₁₀**

2 1 0 A = 10 10 x 16° = 10

**2.- Representa los siguientes números en el formato IEEE754 de precisión simple (desplazamiento = 127).**

**1 bit 8 bits 23 bits**

|  |  |  |
| --- | --- | --- |
| **1** | **10000010** | **11100110011010111000010** |

**S Exp Mantisa**

**0 = + En la posición S (signo) ponemos 1 en este caso porque el número es negativo 🡪 -15,21**

**1 = -**

**a) -15,21 🡪 Empezaremos los cálculos como si fuera positivo**

**15,21 🡪** 1111,0011 (movemos la , al primer 1 de la izquierda, en este caso 3 posiciones por eso multiplicamos por 2 elevado a 3) **🡪** 1**,**1110011 **x** 2ᶟ

A la parte fraccionaria le ponemos en principio sólo 4 números (0011), para no alargarlo demasiado, el resto correspondiente (calculado multiplicando x2 los números fraccionarios y demostrado más abajo), se lo añadimos en la **Mantisa.**

Ahora usamos la potencia 3 para sumarle al desplazamiento 127:

3 + 127 = 130 🡪 10000010

Ahora calculamos los 1´s y 0´s de la parte fraccionaria que van a corresponder a la **Mantisa**

0,21x 2 = 0,42 0,44 x 2 = 0,88 0,16 x 2 = 0,32

0,42 x 2 = 0,84 0.88 x 2 = 1,76 0,32 x 2 = 0,64

0,84 x 2 = 1,68 0,76 x 2 = 1,52 0,64 x 2 = 1,28

0,68 x 2 = 1,36 0,52 x 2 = 1,04 0,28 x 2 = 0,56

0,36 x 2 = 0,72 0,04 x 2 = 0,08

0,72 x 2 = 1,44 0,08 x 2 = 0,16

**b) +31,25 🡪 El signo (S) será 0 porque el número es positivo**

**31,25 🡪** 11111,01 **🡪** 1,111101 **x** 2⁴

4 + 127 = 131 **🡪** 10000011

Ahora tomamos la parte fraccionaria (0,25) del número total (31,25) para hacer los cálculos correspondientes y obtener el valor correcto de la **MANTISA:**

0,25 **x** 2 = 0,50

0,50 **x** 2 = 1 **🡪** A partir de este resultado, no tiene sentido seguir haciendo cálculos, en su caso, usaremos 0´s

**1 bit + 8 bits + 23 bits = 32 bits**

|  |  |  |
| --- | --- | --- |
| **0** | **10000011** | **11110101**000000000000000 |

**S Exp Mantisa**

**3.- Representa los siguientes números en el formato IEEE754 de precisión doble (desplazamiento 1023):**

**a) -15,21**

**b) 31,25**

Realizaremos el ejercicio de la misma forma que el anterior, con la diferencia de que el desplazamiento en este caso será de 1023 bits debido a que la precisión doble (como bien dice el enunciado), en lugar de 32 bits totales, son 64 bits.

Empezamos con el apartado **a)** cuyo número es negativo 🡪 -15,21 :

**-15,21 🡪1111,0011 🡪 1,1110011 x 2ᶟ**

3 + 1023 = 1023 **🡪** 10000000010 **🡪 Exp**

Ahora calculamos la zona fraccionaria (0,21):

0,21 x 2 = 0,42 0,28 x 2 = 0,56 0,04 x 2 = 0,08 0,72 x 2 = 1,44

0,42 x 2 = 0,84 0,56 x 2 = 1,12 0,08 x 2 = 0,16 0,44 x 2 = 0,88

0,84 x 2 = 1,68 0,12 x 2 = 0,24 0,16 x 2 = 0,32 0,88 x 2 = 1,76

0,68 x 2 = 1,36 0,24 x 2 = 0,48 0,32 x 2 = 0,64 0,76 x 2 = 1,52

0,36 x 2 = 0,72 0,48 x 2 = 0,96 0,64 x 2 = 1,28 0,52 x 2 = 1,04

0,72 x 2 = 1,44 0,96 x 2 = 1,92 0,28 x 2 = 0,56 0,04 x 2 = 0,08

0,44 x 2 = 0,88 0,92 x 2 = 1,84 0,56 x 2 = 1,12 0,08 x 2 = 0,16

0,88 x 2 = 1,76 0,84 x 2 = 1,68 0,12 x 2 = 0,24

0,76 x 2 = 1,52 0,68 x 2 = 1,36 0,24 x 2 = 0,48

0,52 x 2 = 1,04 0,36 x 2 = 0,72 0,48 x 2 = 0,96

0,04 x 2 = 0,08 0,72 x 2 = 1,44 0,96 x 2 = 1,92

0,08 x 2 = 0,16 0,44 x 2 = 0,88 0,92 x 2 = 1,84

0,16 x 2 = 0,32 0,88 x 2 = 1,76 0,84 x 2 = 1,68

0,32 x 2 = 0,64 0,76 x 2 = 1,52 0,68 x 2 = 1,36

0,64 x 2 = 1,28 0,52 x 2 = 1,04 0,36 x 2 = 0,72

**1bit + 11 bits + 52 bits = 64bits**

|  |  |  |
| --- | --- | --- |
| **1** | **10000000010** | **1110011**010111000010100011110101110000101000111101011100 |

**S Exp Mantisa**

Ahora empezamos con el apartado **b)** cuyo número **positivo** es 31,25, por lo tanto, **S=0**.

+31,25 🡪 11111,01 🡪 1,**111101** x 2⁴ ; 4 + 1023 = 1027 🡪 **10000000011**

0,25 x 2 = 0,50

0,50 x 2 = 1

**1bit + 11 bits + 52 bits = 64 bits**

|  |  |  |
| --- | --- | --- |
| **0** | **10000000011** | **111101**0000000000000000000000000000000000000000000000 |

**S Exp Mantisa**

**0 = +**

**S**

**1 = -**

**4.- Realiza las siguientes operaciones binarias:**

**a) 1001 1011 🡪** 9 + 11 = 20 🡪 10100

**b) 111010 + 1111 🡪** 58 + 15 = 73 🡪 1001001

**c) 1011 – 0101 🡪** 11 – 5 = 6 🡪 110

**d) 110111 – 10101 🡪** 55 – 21 = 34 🡪 100010

**e) 1110001 x 111 🡪** 113 x 7 = 791 🡪 1100010111

**5.- Un microprocesador de 32 bits cuyas instrucciones de 32 bits están compuestas por dos campos: el primer byte contiene el código de operación (OPCODE) y los restantes, un operando o una dirección de operando.**

**a) ¿Cuál es el tamaño de memoria que se puede direccionar?**

**b)¿Cuántos bits necesita el contador de programa (PC) y el registro de instrucciones**  **(IR)?**

**a)**

32 bits – 8 bits (OPCODE) = 24 bits para direccionar que son 2²⁴ palabras

**b)**

PC = memoria direccionada = 24 bits

IR = Registro de Instrucciones

IR = OPCODE + PC 🡪 IR = 8 + 24 = 32 bits

**6.- Disponemos de un procesador P1 que trabaja a 1 MHz y dedica 5,2 ciclos en realizar cada instrucción (CPI). Otro procesador P2 trabaja a 1,5 MHz y un CPI de 7,3.**

**a) ¿Cuál de los dos tarda menos en realizar un determinado programa?.**

**b) Calcula los MIPS de cada procesador.**

**c) Determina el tiempo que tardará cada procesador en ejecutar un programa con 220000 instrucciones.**

**a)**

1 MHz = 1000000 Hz

**Fórmula de frecuencia en Hz** 🡪  **t = Tiempo (en segundos)**

**Para P1:**

1000000 = => t = => t = 5,2 x 10⁻⁶ s

**Para P2:**

1500000 = => t = => t = 4,86 x 10⁻⁶ s

P2 es más rápido procesando que P1 porque tarda menos tiempo en terminar el mismo proceso y/o cálculo correspondiente.

**b) CHULETA DE FÓRMULAS (FUNDAMENTALES)**

* **MIPS (Millones de Instrucciones Por Segundo)**

**MIPS** x 10⁻⁶ **t = Tiempo (en segundos)**

* **FRECUENCIA (Hz)**

**FRECUENCIA**  **t = Tiempo (en segundos)**

* **TIEMPO DE UN CICLO (s : en segundos)**

**Tpo de un ciclo (s)**

* **CPI (Ciclos Por Instrucción)**

**CPI** =

* **Tiempo medio instrucción**

**Tpo medio instrucción = CPI x Tpo de un Ciclo**

* **OPCODE:** Código de operación, es la porción de una instrucción de lenguaje de máquina que especifica la operación a ser realizada.
* **PALABRA:** Nº de bits 🡪 8, 16, 32 ó 64 Según Arquitectura del CPU.

Una vez reunidas las fórmulas que vamos a necesitar para realizar los cálculos, ahora si, calculamos:

**Para P1:**

**MIPS**  x 10⁻⁶ => x 10⁻⁶ = 0,19230 MIPS

**Para P2:**

**MIPS**  x 10⁻⁶ => x 10⁻⁶ = 0,20576 MIPS

**c) Para calcular el tiempo que tarda cada procesador en ejecutar el programa con 220000 instrucciones:**

**Para P1:**

Si P1 necesita 5,2 Ciclos para 1 Instrucción, para 220000 instrucciones 🡪 5,2 x 220000

5,2 x 220000 = 1144000 Ciclos

Si para 1 Ciclo necesita **Tpo de un ciclo (s)**

**Tpo de un ciclo (s)**  = 1 x 10⁻⁶ s

Para 1144000 Ciclos necesitará:

t = 1144000 x (1 x 10⁻⁶s) = 1,144 s

**Para P2:**

7,3 x 220000 = 1606000 Ciclos

**Tpo de un ciclo (s)**  = 6,6666 x 10⁻⁷ s

Para 1606000 Ciclos necesitará:

1606000 x (6,6666 x 10⁻⁷) = 1,07065 s