
Pregunta	si	no
1. Considere una instrucción **beq** hipotética de 32 bits de longitud cuyo formato es el siguiente: ▪ 6 bits para el **opcode** ▪ 6 bits para codificar el primer registro ▪ 6 bits para codificar el segundo registro ▪ 14 bits para codificar el offset (considerado como un número sin signo) que va sumado al PC en el caso de que el salto se tome Analice el código que se muestra en la Figura 1. ¿Es cierto que este código es incorrecto?	☐	■

Respuesta: el código de la Figura 1 es correcto. Los 14 bits de offset permiten codificar un número cuyo valor máximo es $2^{14} - 1$ (=16383). **Así que, incluso si el PC no ha sido incrementado todavía, la máxima dirección alcanzable es 21384 (5004 + 16383), que es mayor de la dirección 21256.**

2. Asuma ahora que el PC ha sido incrementado de 4 cuando la máquina realiza la comparación relativa a la **beq** en la dirección **5004**. ¿Es cierto que en esta condición el alcance máximo del salto es de 2048 instrucciones?	☐	■

Respuesta:
$\#\ instrucciones = \lfloor (2^{14} - 1)/2^2 \rfloor + 1 = 4095 + 1 = 4096$. **Por tanto, el alcance máximo es superior a 2048 instrucciones. Observe que el operador $\lfloor\ \rfloor$ es necesario porque la dirección debe ser un múltiplo de 4.**

3. En una máquina de tipo *little-endian* la siguiente palabra de 32 bits se halla a partir de la dirección **1000**: **`CD 12 AB 90`**. Si este valor representa un número entero natural, ¿es cierto que su valor decimal sería 2.427.130.573?	■	☐

Respuesta:

```
1000:  CD
1001:  12
1001:  AB
1003:  90
```

La máquina es de tipo *Little endian*, por tanto, el byte más significativo se halla en la dirección más alta (es decir, la `1003`). Esto significa que el número almacenado en memoria es `90AB12CD` = 2.427.130.573

4. A=**10010011** y B=**00111110** representan dos números binarios con signo representados en complemento a 2. ¿Es cierto que el |A|< |B|? ☐ ■

Respuesta:
A es negativo, por tanto, para calcular su módulo tengo que coger el complemento a 2 del número: $|A| = 01101101 > |B| = 00111110$

5. El numero binario natural **10 1111 0111 1010** equivale a **2F7A** en hexadecimal. ¿Es esto cierto? ■ ☐

Respuesta: $\underbrace{10}_{2}\ \underbrace{1111}_{F}\ \underbrace{0111}_{7}\ \underbrace{1010}_{A}$

Dirección	Instrucción
5000	…
5004	beq R12,R11,X
5008	add R1, R2, R3
.....	…
X: 21256	sub R1, R2, R3

Figura 1. Mapa de memoria de un hipotético programa

PREGUNTA 2.1. Considere dos computadores C1 y C2 que implementan el mismo repertorio de instrucciones. El juego de instrucciones se compone de tres clases distintas (A, B, y C). El computador C1 funciona a una frecuencia de reloj de 80 MHz, mientras que C2 funciona con un reloj de 100 MHz. El número medio de ciclos para cada clase de instrucción, así como su frecuencia estadística para un programa típico se muestran en la tabla siguiente:

Clase de Instrucción	Ciclos/Clase Instrucción para C1	Ciclos/Clase Instrucción para C2	Frecuencia
A	1	2	60%
B	2	3	30%
C	4	4	10%

¿Cuál de los dos computadores tiene peor prestación (medida en millones de instrucciones por segundo –MIPS)? Para este computador, ¿Cuál clase de instrucción debería modificar para que éste pueda tener prestaciones mejores o iguales a las del mejor de los dos (sólo puede modificar una sola clase de instrucciones del computador más lento)?

Respuesta:

Para C1: CPI = (60/100)*1 + (30/100)*2 + (10/100)*4 = 1.6

Para C2: CPI = (60/100)*2 + (30/100)*3 + (10/100)*4 = 2.5

sigue que:

Para C1: MIPS = frecuencia de reloj/(CPI * 10^6) = (80 * 10^6) / (1.6*10^6) = 50.0 Para C2: MIPS = frecuencia de reloj/(CPI * 10^6) = (100 * 10^6) / (2.5*10^6) = 40.0

El computador C2 tiene prestaciones inferiores. Si reducimos a 1 el CPI de las instrucciones de clase A para el computador C2 podemos aumentar los MIPS tal y como se muestra a continuación:

CPI = (60/100)*1 + (30/100)*3 + (10/100)*4 = 1.9
MIPS = frecuencia de reloj/(CPI * 10^6) = (100 * 10^6) / (1.9*10^6) = 52.6

PREGUNTA 2.2. Escriba el código ensamblador MIPS equivalente al siguiente fragmento de programa en lenguaje C:

```
int A[100], B[100];
for (i=1; i < 100; i++) {
   A[i] = A[i-1] + B[i];
}
```

Asuma que las direcciones de base de los vectores `A` y `B` se hallen en los registros `$a0` y `$a1` respectivamente.

Respuesta:

```
            li $t0, 1               # valor inicial del contador i
            li $t5, 100             # límite superior del bucle
loop:
            lw $t1, 0($a1)          # cargo A[i-1]
            lw $t2, 4($a2)          # cargo B[i]
            add $t3, $t1, $t2       # A[i-1] + B[i]
            sw $t3, 4($a1)          # A[i] = A[i-1] + B[i]
            addi $a1, $a1, 4        # apunto al próximo elemento de A
            addi $a2, $a2, 4        # apunto al próximo elemento de B
            addi $t0, 1             # incremento el contador i
            bne $t0, $t5, loop      # comparo el Contador con el límite superior
finish:
```

PREGUNTA 2.3. Considere el siguiente fragmento de programa en lenguaje C:

```
int v[10], s;
int *p;
s = 17;
for (p = &v[3]; *p != 0; p++)
   s = s + *p;
```

El siguiente fragmento en lenguaje ensamblador MIPS muestra una **traducción errónea** del código C anterior, en la que se asume que `s` se halla `$s0` en y que el puntero `p` se halla en `$s3`.

```
            or $s0, $zero, $zero
            lw $s3, v+12
loop:
            bne  $t0, finish
            add  $s0,$s3,$s0
            addi $s3, 1
            j loop
finish:
```

En el código ensamblador anterior hay **6** errores (incluido una instrucción que falta). Hállelos y corríjalos.

Respuesta:

```
# Los cambios respecto al código erróneo están subrayados
            li $s0,17
            la $s3,v+12
loop:
            lw $t0,0($s3)
            beq $t0,finish
            add $s0,$t0,$s0
            addi $s3,$s3,4
            j loop
finish:
```

PROBLEMA 3.1. NVIDIA tiene un "medio" formato parecido al IEEE 754 excepto por el hecho que es de 16 bits. El bit más significativo representa el signo, seguido por un exponente sobre 5 bits representado en exceso-16 y una mantisa de 10 bits con *hidden bit*. Escriba el patrón binario relativo al número $X=-1.5625\times10^{-1}$. Compare el rango y la precisión de este formato con rango y precisión del formato IEEE 754 de precisión sencilla.

Finalmente, asuma $A=2.6125\times10^{1}$ y $B=4.150390625\times10^{-1}$. Calcule la suma de A y B **a mano** asumiendo el "medio" formato NVIDIA descrito anteriormente. Asuma un bit de guarda, un bit de redondeo y un *sticky bit*. Redondee al número par más próximo. Ilustre todos los pasos seguidos.

Respuesta:

Problema resuelto en clase el día 24/10/2016. Se pone sólo la operación de suma.

$2.6125 \times 10^{1} + 4.150390625 \times 10^{-1}$
$2.6125 \times 10^{1} = 26.125 = 11010.001 = 1.1010001000 \times 2^{4}$
$4.150390625 \times 10^{-1} = .4150390625 = .011010100111 = 1.1010100111 \times 2^{-2}$
Desplazo el punto decimal de 6 posiciones a la izquierda para alinear los exponentes:

```
             GRS
1.1010001000 000 0000
+.0000011010 101 0111 (Guard = 1, Round = 0, Sticky = 1 hay bits no nulos)
--------------------
1.1010100010 101
```

En este caso los bits extra (G,R,S) tienen un peso de 5×2^{-13} >0.5 LSB que, por tanto, debe ser redondeado:

$1.1010100011 \times 2^4 = 11010.100011 \times 2^0 = 26.546875 = 2.6546875 \times 10^1$