Estructura de Computadores

Tema 5. Aritmética de enteros y coma flotante

(0,75 p) Suposem que els registres MIPS $t1 i $t2 valen $t1=0x00000010 i $t2=0xFFFFFFFF. Indica el contingut final en hexadecimal de $hi i $lo després d'executar `mult $t1,$t2` i després de `multu $t1,$t2`.

```
mult     $t1, $t2     # $hi= 0x          $lo= 0x
multu    $t1, $t2     # $hi= 0x          $lo= 0x
```

Problema

- En la suma de dos naturales se produce desbordamiento (overflow) cuando el resultado es menor que cualquiera de los dos sumandos. Basándote en esta propiedad, haz un programa que, dados dos naturales almacenados en los registros $t1 y $t2, realize la suma y guarde un 1 en $t3 si se ha producido overflow y un 0 en caso contrario. No se pueden usar instrucciones de salto.

Problema

- Dada la siguiente declaración en C:

```
long long x, y;
```

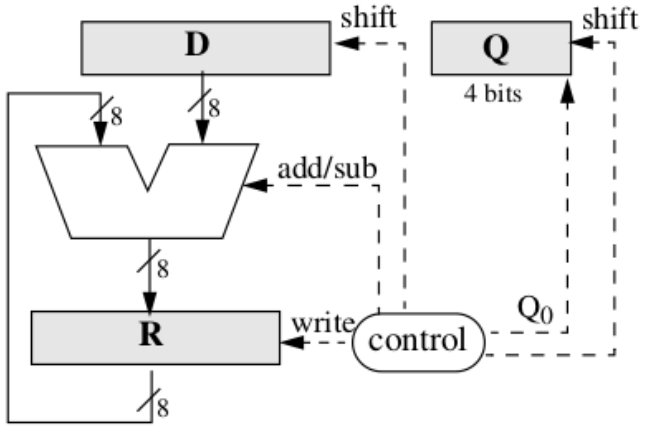
- Traduce a MIPS la siguiente sentencia:

```
x = x + y;
```

b) Suposem que els registres MIPS `$t1` i `$t2` valen `$t1=0x12345678` i `$t2=0xFFFFFFFF`. Indica el contingut final en hexadecimal de `$hi` i `$lo` després d'executar `divu $t1,$t2` i després de `div $t1,$t2`.

divu:	$hi=	0x	$lo=	0x
div:	$hi=	0x	$lo=	0x

Sigui el següent circuit seqüencial per a la divisió de números naturals de 4 bits, anàleg al que s'ha estudiat durant el curs, el qual calcula el quocient i el residu amb 4 bits:



Suposem que volem calcular la divisió (en base 2): 1111/0010. Omple la següent taula indicant quin és el valor inicial dels registres R, D i Q, així com el seu valor després de cada iteració de l'algorisme que controla el circuit (omple tantes files com iteracions tingui l'algorisme).

iteració	R (Dividend/Residu)								D (Divisor)								Q (Quocient)			
valor inicial									0	0	1	0	0	0	0	0				
1																				
2																				

Representación de números en coma flotante

- Sirve para codificar números reales de forma aproximada
- Notación científica (base 10)
  - $234000000 = 234 \times 10^6 = 2,34 \times 10^8 = 0,0234 \times 10^{10}$

Representación de números en coma flotante

- Sirve para codificar números reales de forma aproximada
- Notación científica (base 10)
  - $234000000 = 234 \times 10^6 = 2,34 \times 10^8 = 0,0234 \times 10^{10}$
- Notación científica **normalizada** (base 10)
  - $2,34 \times 10^8$
  - $v = m \times 10^n$
  - m: mantisa con parte entera e y parte fraccionaria f
    - $m = e, f$
    - $0 < e \leq 9$
    - Indica los dígitos significativos de la magnitud
  - n: exponente
    - Indica la posición de la coma (factor de escala)

Representación de números en coma flotante

- Representación binaria
  - *xxx...x*: dígitos binarios de la mantisa
  - *yyy...y*: dígitos binarios del exponente

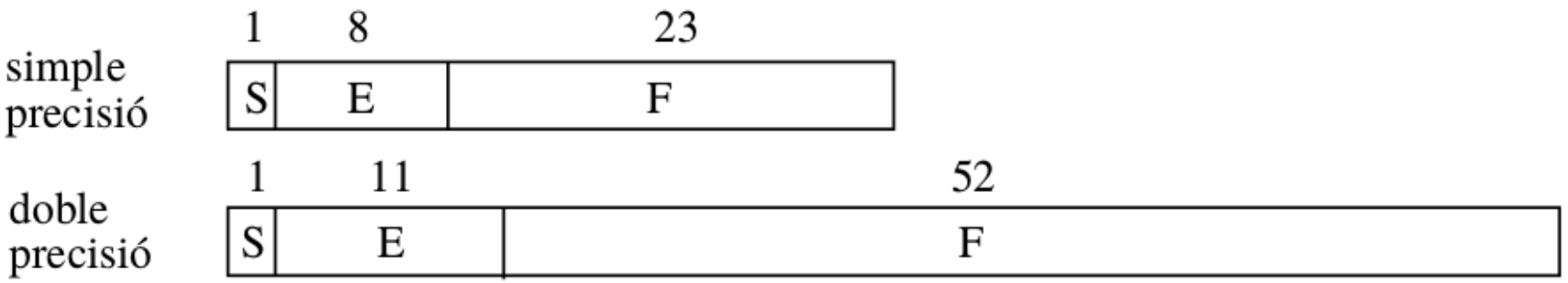
$$v = \pm \overbrace{1,xxx \ldots x}^{\text{fracció(F)}} \times 2 \underbrace{yyy \ldots y}_{\text{exponent(E)}}$$

signe(S) mantissa base

- Se almacenan el signo (S), la fracción (F) y el exponente (E)
- No se almacenan:
  - El dígito entero de la mantisa, siempre es 1 (bit oculto)
  - La base, siempre es 2

El estándar IEEE-754

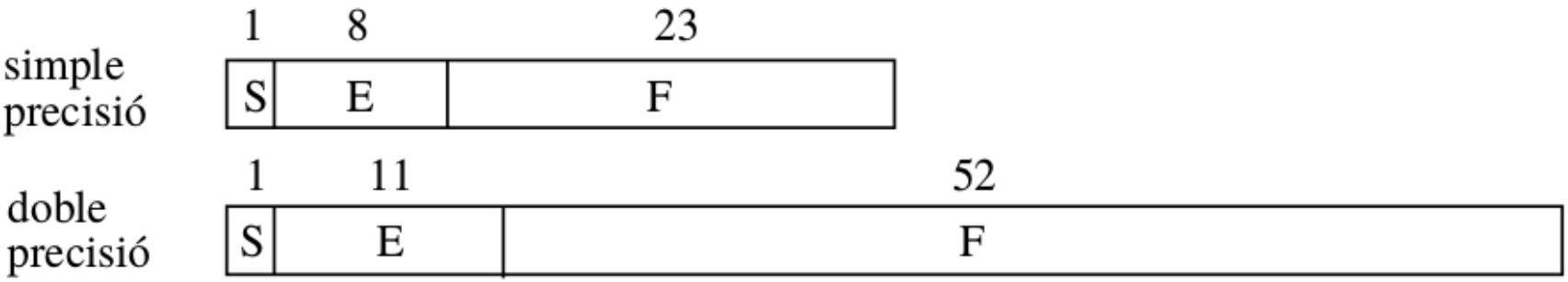
- Define el formato de los números en coma flotante



- **Signo (S)**
  - 0=positivo, 1=negativo
- **Fracción (F)**
  - Parte fraccionaria de la mantisa
- **Exponente (E)**
  - Codificado en exceso 127 (simple precisión) o 1023 (doble precisión)

El estándar IEEE-754

- Define el formato de los números en coma flotante



- Facilita las operaciones de comparación:
  - Si tienen signo distinto, el positivo es el mayor
  - Si tienen el mismo signo, se pueden comparar usando un comparador de naturales
    - Si son negativos, hay que invertir el resultado de la comparación

Rango y precisión

- Rango de representación
  - Intervalo formado por el mayor número positivo representable y su opuesto
  - Mayor número: todos los bits a 1 en la mantisa y máximo exponente (E_{max})
  - La extensión del rango depende principalmente del número de bits del exponente
  - Si $E > E_{max}$ tenemos **overflow**

Rango y precisión

- Rango de representación
  - Intervalo formado por el mayor número positivo representable y su opuesto
  - Mayor número: todos los bits a 1 en la mantisa y máximo exponente (E_{max})
  - La extensión del rango depende principalmente del número de bits del exponente
  - Si $E > E_{max}$ tenemos **overflow**
- Precisión
  - Número de bits significativos
  - Grado de detalle con el que se puede representar una cantidad
  - Igual al número de bits de la mantisa

Error de precisión

- Error cometido debido a que la mantisa tiene un número limitado de bits
  - Sea v un valor no representable

Error de precisión

- Error cometido debido a que la mantisa tiene un número limitado de bits
  - Sea v un valor no representable
  - Sea $v_0 < v < v_1$

Error de precisión

- Error cometido debido a que la mantisa tiene un número limitado de bits
  - Sea v un valor no representable
  - Sea $v_0 < v < v_1$
  - Si aproximamos v usando v_0 el error es: $\epsilon = |v - v_0|$

Error de precisión

- Error cometido debido a que la mantisa tiene un número limitado de bits
  - Sea v un valor no representable
  - Sea $v_0 < v < v_1$
  - Si aproximamos v usando v_0 el error es: $\epsilon = |v - v_0|$
  - Máximo **error absoluto**: $\epsilon_{max} < |v_1 - v_0|$

Error de precisión

- Error cometido debido a que la mantisa tiene un número limitado de bits
  - Sea v un valor no representable
  - Sea $v_0 < v < v_1$
  - Si aproximamos v usando v_0 el error es: $\epsilon = |v - v_0|$
  - Máximo **error absoluto**: $\epsilon_{max} < |v_1 - v_0|$
  - En simple precisión:
    - $v_0 = m \times 2^E$
    - $v_1 = (m + 2^{-23}) \times 2^E$
    - $\epsilon_{max} = 2^{E-23}$

Error de precisión

- Error cometido debido a que la mantisa tiene un número limitado de bits
  - Sea v un valor no representable
  - Sea $v_0 < v < v_1$
  - Si aproximamos v usando v_0 el error es: $\epsilon = |v - v_0|$
  - Máximo **error absoluto**: $\epsilon_{max} < |v_1 - v_0|$
  - En simple precisión:
    - $v_0 = m \times 2^E$
    - $v_1 = (m + 2^{-23}) \times 2^E$
    - $\epsilon_{max} = 2^{E-23}$
  - Error relativo: $\eta = \epsilon / |v|$

Error de precisión

- Error cometido debido a que la mantisa tiene un número limitado de bits
  - Sea v un valor no representable
  - Sea $v_0 < v < v_1$
  - Si aproximamos v usando v_0 el error es: $\epsilon = |v - v_0|$
  - Máximo **error absoluto**: $\epsilon_{max} < |v_1 - v_0|$
  - En simple precisión:
    - $v_0 = m \times 2^E$
    - $v_1 = (m + 2^{-23}) \times 2^E$
    - $\epsilon_{max} = 2^{E-23}$
  - Error relativo: $\eta = \epsilon / |v|$
  - Máximo **error relativo** en simple precisión:
    - $\eta_{max} < \epsilon_{max} / |v_0|$
    - $\eta_{max} < 2^{E-23} / (m \times 2^E) = 2^{-23} / m$
    - Como $m \geq 1{,}0$: $\eta_{max} < 2^{-23}$

Redondeo

- El resultado exacto v de una operación puede no ser representable
  - La mantisa tiene un número limitado de bits (precisión)
- Hay que aproximar v por uno de los dos valores más próximos, v_0 o v_1, siendo $v_0 < v < v_1$
- Esta aproximación recibe el nombre de redondeo
- Cuatro modos de redondeo en IEEE-754 para elegir entre v_0 y v_1
  - El más cercano a v (modo utilizado por defecto)
  - El más cercano a cero
  - El más cercano a +Infinito
  - El más cercano a -Infinito

Redondeo al más cercano a v

- Proporciona el menor error de precisión
  - $\epsilon_{max} < |v_1 - v_0|/2$
  - $\eta_{max} < 0,5 \times 2^{-23}$
  - $\eta_{max} < 2^{-24}$
- Supongamos que tenemos resultados con 6 bits que queremos redondear a 3 bits:
  - 101,010 $\rightarrow$ 101 (en decimal: 5.250 $\rightarrow$ 5)
  - 101,011 $\rightarrow$ 101 (en decimal: 5.375 $\rightarrow$ 5)
  - 101,101 $\rightarrow$ 110 (en decimal: 5.625 $\rightarrow$ 6)
  - 101,100 $\rightarrow$? (en decimal: 5.500 $\rightarrow$?)

Redondeo al más cercano a v

- Proporciona el menor error de precisión
  - $\epsilon_{max} < |v_1 - v_0|/2$
  - $\eta_{max} < 0,5 \times 2^{-23}$
  - $\eta_{max} < 2^{-24}$
- Supongamos que tenemos resultados con 6 bits que queremos redondear a 3 bits:
  - 101,010 $\rightarrow$ 101 (en decimal: 5.250 $\rightarrow$ 5)
  - 101,011 $\rightarrow$ 101 (en decimal: 5.375 $\rightarrow$ 5)
  - 101,101 $\rightarrow$ 110 (en decimal: 5.625 $\rightarrow$ 6)
  - 101,100 $\rightarrow$? (en decimal: 5.500 $\rightarrow$?)
- En caso de equidistancia, se redondea al número par:
  - 101,100 $\rightarrow$ 110 (en decimal: 5,500 $\rightarrow$ 6)

Redondeo en IEEE-754 - Resumen

- Redondear al más próximo a v
  - Da el menor error
  - Se utiliza por defecto
  - Redondeo al número par en caso de equidistancia
- Redondear al más próximo a cero
  - Truncar los bits que sobran
  - Fácil de implementar
- Redondear al más próximo a +Infinito
- Redondear al más próximo a -Infinito

Underflow

- Se produce cuando $|v|$ es menor que el mínimo positivo representable
  - $0 < |v| < 2^{E_{min}}$
  - Si aproximamos v por 0:
    - Error absoluto: $\epsilon = |v - 0|$
    - Error relativo: $\eta = \epsilon / |v| = 1$
  - El error relativo en el intervalo $(-2^{E_{min}}, 2^{E_{min}})$ es 6 órdenes de magnitud mayor que en el resto del rango

Codificaciones especiales

- Se reservan dos codificaciones del exponente para valores especiales:
  - E=000...0 (todo ceros)
  - E=111...1 (todo unos)
- E_{min} = 00...001 y E_{max} = 111..110
  - Precisión simple: $[-126, +127]$
  - Precisión doble: $[-1022, +1023]$

Codificaciones especiales - Denormales

- Error relativo enorme en el intervalo $(-2^{E_{min}}, 2^{E_{min}})$
- Para reducir el error, se permiten valores denormalizados
- Los denormales tienen la forma: $v = 0, xxx...x \times 2^{E_{min}}$
- Para los denormales, el bit oculto es cero
- Mejor aproximación para valores muy próximos a cero
- Representación de los denormales:
  - Exponente con todos los bits a cero y mantisa con al menos un bit no nulo
  - E=000...0, F=xxx...x (no todos son cero)

Codificaciones especiales - Cero

- Representación del cero
  - Todos los bits del exponente y la fracción a cero
  - F=000...0 y E=000...0
  - Debido al signo, el cero tiene dos representaciones

Codificaciones especiales - Infinito

- Resulta conveniente operar con el infinito

$$y = \frac{1}{1 + \frac{100}{x}}$$

- 1/0=Inf
- 1/Inf=0
- x+Inf=Inf
- Representación del infinito
  - Todos los bits de la fracción a cero y los del exponente a uno
  - F=000...0, E=111...1
  - Con el bit de signo tenemos +Infinito y -Infinito

Codificaciones especiales - Not a Number (NaN)

- Resultados inválidos (NO confundir con overflow/underflow)
  - Raíz cuadrada de un número negativo
  - Logaritmo de un número negativo
  - 0 / 0
  - 0 x Inf
  - Inf / Inf
- Representación de NaN
  - Todos los bits del exponente a uno, mantisa distinta de cero
  - E=111...1, F=xxx...x (no todos son cero)
- Propagación de NaN
  - Cualquier operación en la que uno de los operandos es NaN da como resultado NaN