DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (I/2026)

Interrogación 2 - Parte 2 - Sección 2

Pauta de evaluación

Pregunta 2: Representación de Números Reales (6 ptos.)

(a) (2 ptos.) Indique cuántos exponentes distintos se pueden representar en un número float del estándar IEEE754 y explique por qué.

Solución: Como los exponentes 00000000b y 11111111b se reservan para las representaciones de 0 y $\pm\infty$/`NaN` respectivamente, el rango de exponentes de los floats bajo el estándar IEEE754 corresponde a $[-126, 127]$, es decir, se tienen 254 exponentes distintos (256 combinaciones distintas - 2 reservadas). Se otorga **1 pto.** por indicar correctamente la cantidad de exponentes distintos, y **1 pto.** por justificación.

(b) (4 ptos.) En los siguientes incisos, se le entregará un par de números reales A, B en formato float del estándar IEEE754 con notación hexadecimal. En cada caso, debe interpretar sus valores y realizar las operaciones de suma o multiplicación solicitadas. Luego, debe indicar el valor del resultado de la operación y señalar si se pierde precisión al almacenarlo de vuelta como float del estándar IEEE754, indicando además cómo queda almacenado en este formato. Los valores deben presentarse en base binaria con notación científica normalizada.

- (2 ptos.) $A + B$. A = 0x58800000, B = 0x3F800000.

Solución:

- A = 0 10110001 0000000 00000000000000000b.
  - **Signo:** 0b $\rightarrow$ positivo.
  - **Exponente:** 10110001b $\rightarrow$ 177 - 127 = 50.
  - **Significante:** 00000000000000000000000b $\rightarrow$ 1,0b.
  - **Valor:** $(1,0 \times 10^{50})$b = 10000..,000b = 2^{50}.
- B = 0 01111111 00000000000000000000000b.
  - **Signo:** 0b $\rightarrow$ positivo.

- **Exponente:** 01111111b → 127 - 127 = 0.
- **Significante:** 00000000000000000000000b → 1,0b.
- **Valor:** $(1,0 \times 10^0)$b = 1b = 1.

De lo anterior, se deduce que la operación a realizar es $2^{50} + 1$. En este caso, no se evaluará la forma en la que se realice la suma; solo que su resultado sea correcto. Aquí se opta por seguir el algoritmo visto en clases: se igualan los exponentes y B se expresa como $0,0000..,001 \times 10^{50}$. Luego, sumando los significantes, se tiene $A + B = (1,0000..,001 \times 10^{50})$b. Finalmente, si se almacena como `float`, se tiene el valor 01011000100000000000000000000000b = 0x58800000 = 2^{50}. **Se pierden los 27 bits menos significantes que aportaban la magnitud de 1 y, por ende, se pierde precisión**.

Se otorgan **0.5 ptos.** por la interpretación correcta de A; **0.5 ptos.** por la interpretación correcta de B; **0.5 ptos.** por señalar el resultado correcto de la operación; y **0.5 ptos.** por indicar correctamente si se pierde precisión o no. Se acepta escribir el valor en potencias de 2.

- (2 ptos.) $A \times B$. A = 0xBF800000, B = 0x7F800000.

Solución:

- A = 1 01111111 00000000000000000000000b.
  - **Signo:** 1b → negativo.
  - **Exponente:** 01111111b → 127 - 127 = 0.
  - **Significante:** 00000000000000000000000b → 1,0b.
  - **Valor:** $-(1,0 \times 10^0)$b = −1b = −1.
- B = 01111111100000000000000000000000b.
  - **Signo:** 0b → positivo.
  - **Exponente:** 11111111b → **reservado para infinito y `NaN`**
  - **Significante:** 00000000000000000000000b → 1,0b.
  - **Valor:** ∞.

De lo anterior, se deduce que la operación a realizar es igual a $-1 \times \infty$. En este caso, no se evaluará la forma en la que se realice la multiplicación; solo que su resultado sea correcto. No obstante, **no es necesario realizarla para deducir que el resultado es igual a** $-\infty$, cuya representación en float es igual a 11111111100000000000000000000000b = 0xFF800000. Por lo tanto, no hay pérdida de precisión al almacenar el valor de vuelta como float.

Se otorgan **0.5 ptos.** por la interpretación correcta de A; **0.5 ptos.** por la interpretación correcta de B; **0.5 ptos.** por señalar el resultado correcto de la operación; y **0.5 ptos.** por indicar correctamente si se pierde precisión o no.

Pregunta 3: Comunicación con dispositivos I/O (6 ptos.)

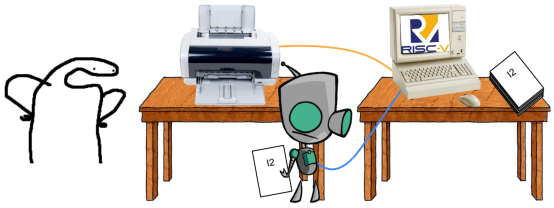
(a) (1.5 ptos.) ¿Qué pasaría, en el contexto de la comunicación con dispositivos I/O a través de interrupciones, si se eliminara la *flag* IF? Justifique su respuesta.

> **Solución:** La *Interruption Flag* le indica a la CPU si puede o no atender interrupciones. Si se elimina, la CPU no contaría con un mecanismo para determinar si debe detener su ejecución y atender una interrupción, impidiendo así la comunicación vía interrupciones. Otro posible escenario sería que se atiendan interrupciones aunque ya haya una en ejecución. Esto implicaría que no haya control en la ejecución anidada de ISRs y que se generen retrasos en su término (o incluso *stack overflow*). Se otorgan **0.75 ptos.** por señalar un posible escenario al eliminar la *flag*, y **0.75 ptos.** por justificación.

(b) (1.5 ptos.) ¿Cuál es la función del vector de interrupciones? ¿Cuál es su contenido?

> **Solución:** La función del vector de interrupciones es permitir la ejecución correcta de la ISR (*Interruption Service Routine*) a partir del identificador del dispositivo a ser atendido, lo que es gestionado por el controlador de interrupciones. Este vector contiene los **punteros** de las ISR, por lo que si se desea atender al dispositivo `i`, se ejecuta el llamado de la subrutina a partir de su dirección igual a `vector_interrupciones[i]`. Se otorgan **0.75 ptos.** por señalar correctamente la función del vector de interrupciones; y **0.75 ptos.** por indicar de forma correcta su contenido.

(c) (3 ptos.) Suponga que es el/la docente del curso y debe imprimir la I2. No obstante, la redacta a última hora y no alcanza a usar los equipos del DCC, por lo que decide imprimir en su hogar con un sistema innovador: una impresora y un robot conectados a un computador con ISA RISC-V. A continuación, una representación gráfica de su implementación:



Para llevar a cabo la comunicación, los registros de 32 bits de los dispositivos se mapean en memoria desde la dirección 0x00FF10AD:

Offset	Nombre	Descripción
0x00	`printer_status`	Registro de estado impresora
0x04	`printer_command`	Registro de comando impresora
0x08	`printer_copies`	Configuración de cantidad de copias impresora
0x0C	`printer_color`	Configuración de color impresora
0x10	`robot_status`	Registro de estado robot
0x14	`robot_command`	Registro de comando robot

Nombre	Descripción	Valor
`printer_status`	Apagado	0
`printer_status`	Encendido	1
`printer_status`	Imprimiendo	2
`robot_status`	Apagado	0
`robot_status`	Encendido	1
`robot_status`	Trabajando	2
`printer_command`	Encender	1
`printer_command`	Apagar	255
`printer_command`	Imprimir	128
`printer_color`	Blanco y negro	1
`printer_color`	Color	2
`robot_command`	Encender	1
`robot_command`	Rellenar papeles impresora	2
`robot_command`	Rellenar tinta impresora	3
`robot_command`	Apilar impresiones*	4
`robot_command`	Apagar	255

* Esta acción hace que el robot pase a estado “Trabajando”, el resto son instantáneas y no cambian su estado.

Para automatizar la impresión, desarrolla el siguiente programa:

```
printing_system:
# Carga 0 + 0x00FF10AD en t0
addi t0, zero, 0x00FF10AD
addi s1, zero, 1
addi s2, zero, 2
addi s3, zero, 3
addi s4, zero, 4
addi s5, zero, 128
addi s6, zero, 255
addi s7, zero, 60

step_one:
    lw t1, 16(t0) # Carga Mem[t0 + 16] en t1
    # Salta a continue_step_one si t1 != zero
    bne t1, zero, continue_step_one
    sw s1, 20(t0) # Carga s1 en Mem[t0 + 20]
    continue_step_one:
        lw t1, 16(t0)
        # Salta a continue_step_one si t1 == s2
        beq t1, s2, continue_step_one

step_two:
    lw t1, 0(t0)
    bne t1, zero, continue_step_two
    sw s1, 4(t0) # ---> Continúa a la derecha
```

```
# --> Continúa aquí
    continue_step_two:
        lw t1, 0(t0)
        beq t1, s2, continue_step_two

step_three:
    sw s7, 8(t0)
    sw s3, 20(t0)
    sw s1, 12(t0)

step_four:
    sw s5, 4(t0)
    continue_step_four:
        lw t1, 0(t0)
        beq t1, s2, continue_step_four

step_five:
    sw s4, 20(t0)
    continue_step_five:
        lw t1, 16(t0)
        beq t1, s2, continue_step_five

step_six:
    sw s6, 4(t0)
    sw s6, 20(t0) # Término del programa.
```

Describa, basándose en las tablas provistas, las tareas que realiza el programa en cada *label* `step_*`. Haga una descripción según los estados y comandos; no una explicación por cada instrucción del programa.

Solución:

- `step_one` **(0.6 ptos.)**: Se revisa el estado del robot. Si está apagado, se enciende antes de seguir. Luego, si está trabajando, se revisa su estado hasta verificar que terminó de trabajar.
- `step_two` **(0.6 ptos.)**: Se revisa el estado de la impresora. Si está apagada, se enciende antes de seguir. Luego, si está imprimiendo, se revisa su estado hasta verificar que terminó de imprimir.
- `step_three` **(0.4 ptos.)**: Se configura un total de 60 copias de impresión; se configura la impresión en blanco y negro; y el robot rellena las tintas de la impresora.
- `step_four` **(0.5 ptos.)**: Se inicia la impresión y se revisa el estado de la impresora hasta que termine de imprimir.
- `step_five` **(0.5 ptos.)**: Se le instruye al robot que apile las impresiones y se espera a que termine de apilarlas.
- `step_six` **(0.4 ptos.)**: Se apaga tanto la impresora como el robot y termina el programa.

Para cada `step`, se otorga la mitad del puntaje si existe como máximo un error o detalle faltante. Si existen dos o más errores o detalles faltantes, no se otorga puntaje.