



IIC2343 - Arquitectura de Computadores (II/2025)

Ayudantía 10

Ayudantes: Daniela Ríos (danielaarp@uc.cl), Alberto Maturana (alberto.maturana@uc.cl), José Mendoza (jfmendoza@uc.cl), Tomás López Massaro (tomas.lopezm20@uc.cl)

Pregunta 1: Preguntas Conceptuales

- (a) Explique el uso e importancia de la señal **IRQ** enviada por el controlador de interrupciones a la CPU.
- (b) ¿Con qué tipo de dispositivos de I/O es preferible utilizar *mapeo* de memoria por sobre puertos?
- (c) Indique cómo se lleva a cabo la transferencia de datos entre dispositivos I/O y la memoria **RAM** en arquitecturas que **carecen** de un controlador de DMA.
- (d) Indique una consecuencia que podría existir si no se respaldan las *flags* computadas por el procesador **antes** de atender la interrupción de un dispositivo I/O en un instante de tiempo determinado.

Pregunta 2: Comunicación con dispositivos I/O

Suponga que decide, con el propósito de ahorrar tiempo, automatizar la preparación de café filtrado a partir de un molinillo y hervidor eléctricos conectados a un computador con ISA RISC-V. Para la comunicación con los electrodomésticos, sus registros de 32 bits son mapeados en memoria desde la dirección 0x600DCAFE:

Offset	Nombre	Descripción
0x00	coffee_grinder_status	Registro estado molinillo
0x04	coffee_grinder_command	Registro comando molinillo
0x08	coffee_grinder_grind_size	Config. grosor molienda
0x0C	coffee_grinder_weight	Config. gramos a moler
0x10	kettle_status	Registro estado hervidor
0x14	kettle_command	Registro comando hervidor
0x18	kettle_temperature	Config. temperatura hervidor (°C)

Nombre	Descripción	Valor
coffee_grinder_status	Apagado	0
coffee_grinder_status	Encendido	1
coffee_grinder_status	Moliendo	2
kettle_status	Apagado	0
kettle_status	Encendido	1
kettle_status	Hirviendo	2
coffee_grinder_command	Encender	1
coffee_grinder_command	Apagar	255
coffee_grinder_command	Moler	128
kettle_command	Encender	1
kettle_command	Apagar	255
kettle_command	Hervir	128
coffee_grinder_grind_size	Fino	1
coffee_grinder_grind_size	Medio	2
coffee_grinder_grind_size	Grueso	3

Pensando en su preparación favorita, desarrolla el siguiente programa:

```
coffee_system:
# Carga 0 + 0x600DCAFE en t0
addi t0, zero, 0x600DCAFE
addi t2, zero, 0
addi t3, zero, 0
addi s1, zero, 1
addi s2, zero, 2
addi s3, zero, 128
addi s4, zero, 255
addi s5, zero, 30
addi s6, zero, 92

step_one:
    lw t1, 0(t0) # Carga Mem[t0 + 0] en t1
    # Salta a continue_step_one si t1 != s2
    bne t1, s2, continue_step_one
    jal ra, step_five # Llamado a step_five
    beq zero, zero, step_three
    continue_step_one:
        beq t1, s1, step_two
        sw s1, 4(t0) # Carga s1 en Mem[t0 + 4]
    step_two:
        sw s2, 8(t0)
        sw s5, 12(t0)
        sw s3, 4(t0)
        jal ra, step_five
    # Continúa en step_three -->
```

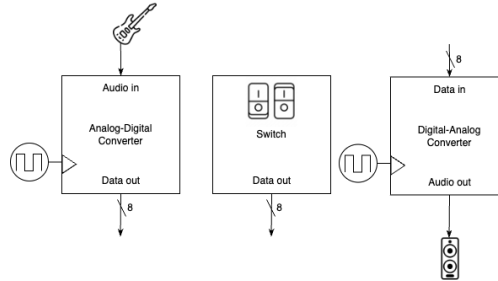
```
step_three:
    lw t1, 16(t0) # Carga Mem[t0 + 16] en t1
    # Salta a continue_step_three si t1 != s2
    bne t1, s2, continue_step_three
    jal ra, step_six
    beq zero, zero, end # Salto incondicional
    continue_step_three:
        beq t1, s1, step_four
        sw s1, 20(t0)
    step_four:
        sw s6, 24(t0)
        sw s3, 20(t0)
        jal ra, step_six
        beq zero, zero, end
    step_five:
        while_step_five:
            lw t1, 0(t0)
            beq t1, s2, while_step_five
            sw s4, 4(t0)
            jalr zero, 0(ra) # Retorno
    step_six:
        while_step_six:
            lw t1, 16(t0)
            beq t1, s2, while_step_six
            sw s4, 20(t0)
            jalr zero, 0(ra)
    end: # 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.

Pregunta 3: Computador básico e I/O

Dado su interés por la música y la tecnología, ha empezado a explorar el funcionamiento de los pedales de efecto haciendo uso del computador básico para modificar las señales de sonido emitidas por una guitarra. Para esto, habilita la conexión con los siguientes dispositivos *port-mapped*:

Dispositivo	Tipo	Puerto
Conversor análogo-digital	Entrada	0
<i>Switch</i> de efecto	Entrada	1
Conversor digital-análogo	Salida	2



El conversor análogo-digital envía la digitalización de la señal generada por la guitarra; mientras que el conversor digital-análogo recibe la señal digital con efectos aplicados por el computador, los que son finalmente transmitidos a un amplificador. Adicionalmente, añade un *switch* con señal de salida de 8 bits, pero solo con dos valores posibles: 0 si está apagado y 1 si está prendido. El efecto del pedal se aplica solo si el *switch* está encendido. Para lograr este comportamiento, debe modificar la microarquitectura del computador básico implementando las siguientes instrucciones:

Instrucción	Operandos	Operación
IN	A,Lit B,Lit	A = Dispositivo asociado a puerto Lit B = Dispositivo asociado a puerto Lit
OUT	Lit,A Lit,B	Dispositivo asociado a puerto Lit = A Dispositivo asociado a puerto Lit = B

Para cada instrucción nueva, deberá incluir la combinación **completa** de señales que la ejecutan. Por cada señal de carga/escritura/incremento/decremento, deberá indicar si se activan (1) o no (0); en las señales de selección, deberá indicar el **nombre** de la entrada escogida (“-” si no afecta). Puede realizar todas las modificaciones en un solo diagrama. Para el conversor digital-análogo, asuma que si no se le envían datos mediante OUT, debe recibir un 0 por defecto.

- IN Reg, Lit. Guarda en Reg (A/B) el valor de salida del puerto Lit.
- OUT Lit, Reg. Entrega como entrada al puerto Lit el valor de Reg (A/B).
- Asuma que se implementan las instrucciones anteriores. Elabore un fragmento de código Assembly que aplique el efecto del pedal a través del llamado a una subrutina `apply_effect`. Esta asumirá que la señal de sonido se encontrará almacenada en B y modificará su valor directamente. El programa debe, entonces:
 - Obtener el valor de la señal del conversor análogo-digital y el estado del *switch*.
 - Aplicar el efecto a partir del llamado a `apply_effect` solo si el *switch* está activado.
 - Enviar al conversor digital-análogo la señal resultante.

Debe hacer uso de las instrucciones IN y OUT para cumplir lo anterior. Puede agregar instrucciones adicionales ya existentes en la ISA del computador básico para completar el programa. **Solo debe llamar a `apply_effect`**, no es necesario que defina su código.

Feedback ayudantía

Escanee el QR para entregar feedback sobre la ayudantía.

