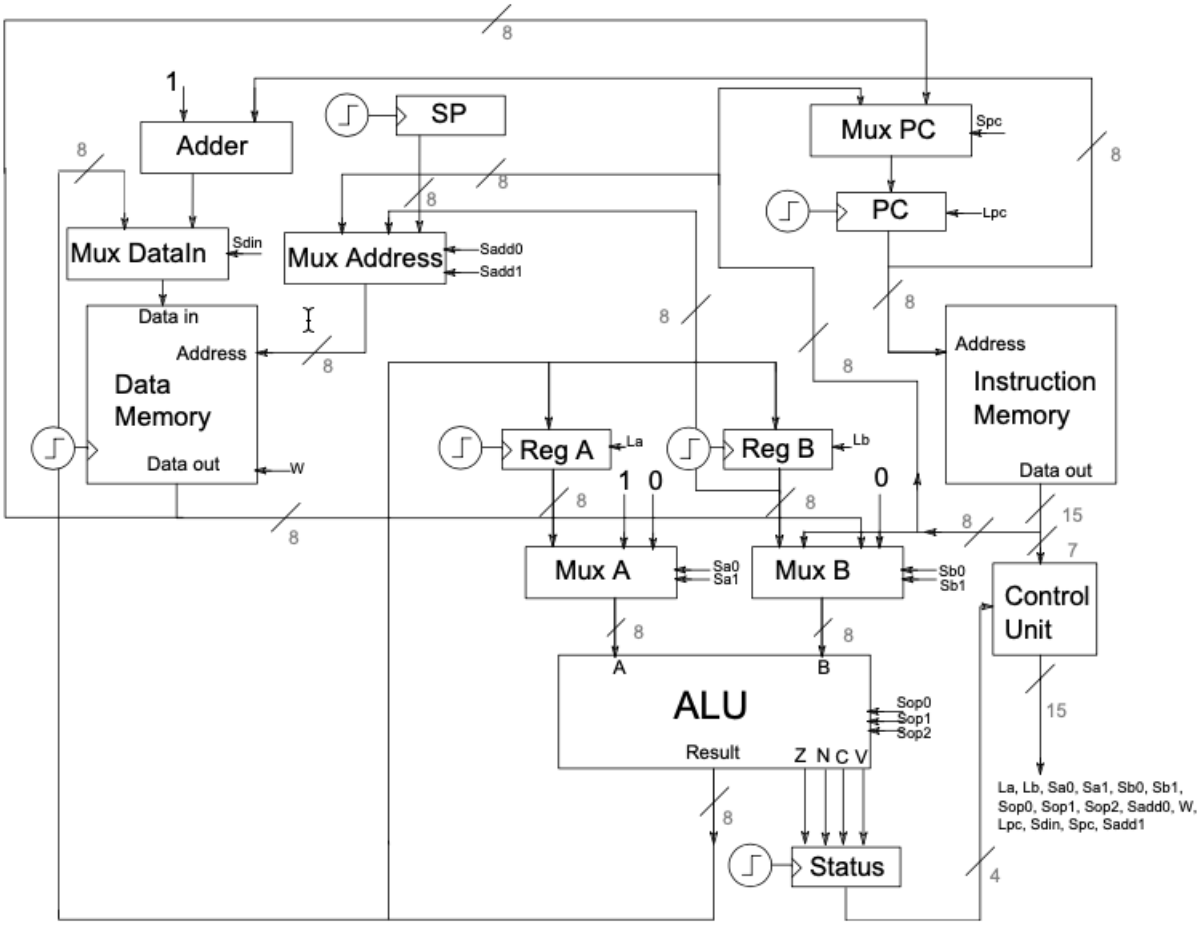
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (II/2025)

Ayudantía 6

Ayudantes: Daniela Ríos (danielaarp@uc.cl), Alberto Maturana (alberto.maturana@uc.cl), Mario Rojas (mario.denzel@estudiante.uc.cl)

Computador básico



Pregunta 1: Subrutinas - Modificación de código

Imagina que te encuentras trabajando como programador(a) en la oficina del famoso juego desarrollado completamente en Assembly: *Rollercoaster Tycoon*. Luego de unas horas, te das cuenta que la letra T del teclado de tu computador ha dejado de funcionar, la que es de suma importancia al momento de escribir subrutinas por el uso de la instrucción `RET`. Adicionalmente, no cuentas con la opción de copiar y pegar texto, por lo que deberá desistir de su uso. Por el motivo anterior, se diseñan las siguientes tres nuevas instrucciones que te permitan simular parte del comportamiento de la instrucción `RET`.

1. `MOV B, SP`: Almacena el valor del registro `SP` en el registro `B`.
2. `JMP B`: Salta a la instrucción con dirección igual al valor almacenado en el registro `B`.
3. `JMP (B)`: Salta a la instrucción con dirección igual al valor almacenado en la memoria de datos en la dirección `B`.

Asuma que se implementan las instrucciones anteriores y modifique el siguiente fragmento de código de forma que no utilice la instrucción `RET`, pero que se llegue al mismo resultado. Luego, comente sobre su resultado: ¿Cumple la misma función que el original? ¿Existe alguna consideración a tener en cuenta?

```
DATA:
    number_of_rides 0

CODE:
    JMP main

    // Incrementa la cantidad de veces a la que se sube a la montaña rusa
    increase_number_of_rides:
        MOV A, (number_of_rides)
        ADD A, 1
        MOV (number_of_rides), A
        RET

    main:
        PUSH A
        CALL increase_number_of_rides
        POP A
```

Pregunta 2: Subrutinas - Computador básico

Ha sido contratado/a como programador/a en una bolsa de valores internacional. Al revisar la infraestructura tecnológica, nota que todos los procesadores se basan en el computador básico del curso, lo que limita el rendimiento necesario para reaccionar ante cambios del mercado. Su misión es diseñar una versión mejorada, llamada *FastBasicComputer*, que permita ejecutar programas con mayor velocidad y eficiencia. Para ello, deberá incorporar **instrucciones nuevas que disminuyan los ciclos de ejecución**. Deberá modificar la microarquitectura del computador básico para implementar las instrucciones nuevas y su funcionamiento. 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.

(a) Implemente la instrucción `CALLFAST label`: Almacena `PC+1` en un **nuevo** registro llamado *Return Address Register* y **no modifica** el *stack*. Además, salta a la dirección `label`.

(b) Implemente la instrucción `RETFAST`: Carga en el contador `PC` el valor almacenado en el registro *Return Address Register* y **no modifica** el *stack*.

(c) Implemente la instrucción `JEQFAST A, Lit, label`: Ejecuta la operación `A - Lit` en la ALU y salta a la dirección asociada a `label` si, y solo si la *flag* `Z` **computada en el mismo ciclo** es igual a 1.

(d) Para evaluar la velocidad de *FastBasicComputer*, se realiza una comparativa entre un programa escrito en el *Assembly* del computador básico `(a)` y otro equivalente en la nueva arquitectura `(b)`. Indique los valores de los registros `A` y `B` al finalizar la ejecución del código en cada caso junto con la cantidad de ciclos de ejecución. Asuma que no se agregan instrucciones para inicializar la memoria RAM.

(a)

```
DATA:
    n 8
    is_even 0
CODE:
    MOV A, (n)
    CALL check_is_even
    JMP end
    check_is_even:
        MOV B, 1
        AND A, B
        CMP A, 0
        JEQ set_is_even
        end_check_is_even:
            RET
        set_is_even:
            MOV (is_even), B
            JMP end_check_is_even
    end:
        MOV A, (is_even)
```

(b)

```
DATA:
    n 8
    is_even 0
CODE:
    MOV A, (n)
    CALLFAST check_is_even
    JMP end
    check_is_even:
        MOV B, 1
        AND A, B
        JEQFAST A, 0, set_is_even
        end_check_is_even:
            RETFAST
        set_is_even:
            MOV (is_even), B
            JMP end_check_is_even
    end:
        MOV A, (is_even)
```

(e) Las instrucciones `CALLFAST label` y `RETFAST` presentan un problema al ejecutar llamados anidados, situación que **no ocurre** con `CALL` y `RET`. En particular, se observa que no se realiza de forma correcta el retorno. Explique por qué ocurre esto.

Feedback ayudantía

Escanee el QR para entregar feedback sobre la ayudantía.