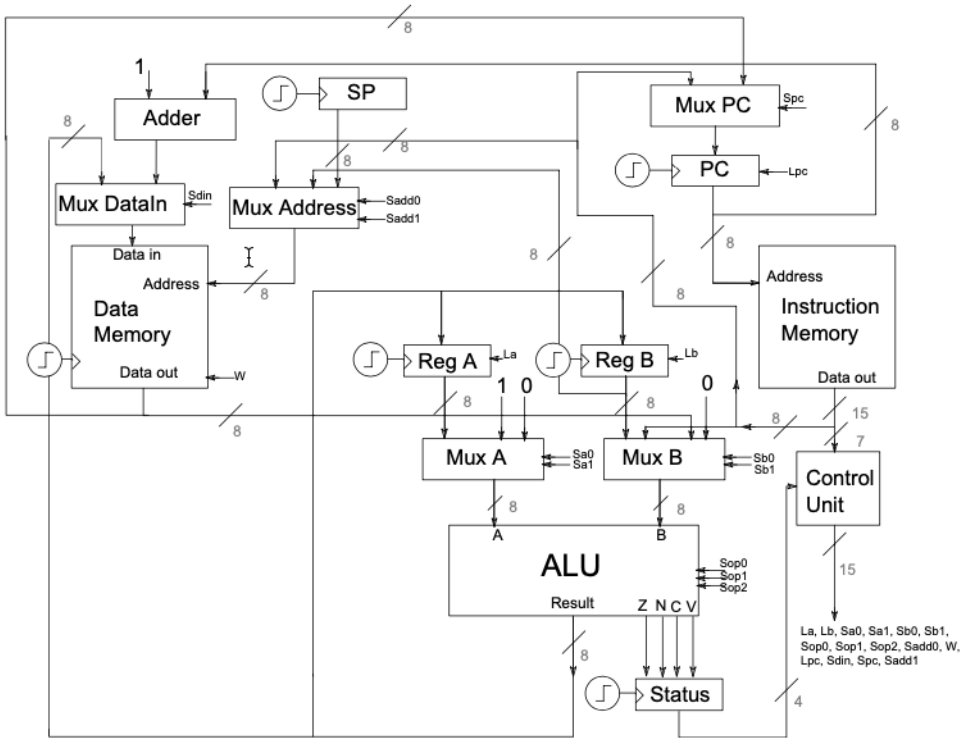
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (II/2025)

Guía de ejercicios: Subrutinas

Ayudantes: Daniela Ríos (danielaarp@uc.cl), Alberto Maturana (alberto.maturana@uc.cl), Nicolás Romo (nroma@uc.cl)

Computador básico con subrutinas



Pregunta 1: Computador básico (P4-I1-2024-1)

Asuma que se encuentra trabajando como programador(a) en la oficina del famoso juego desarrollado completamente en Assembly: *Rollercoaster Tycoon*. Luego de unas horas, se da cuenta que la letra T del teclado de su 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 cuenta con la opción de copiar y pegar texto, por lo que deberá desistir de su uso. Por el motivo anterior, deberá diseñar tres nuevas instrucciones que le 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`.

Deberá modificar la microarquitectura del computador básico para implementar las nuevas instrucciones 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.

Instrucción	L_A	L_B	L_{PC}	W	Inc_{SP}	Dec_{SP}	S_A	S_B	S_{OP}	S_{Add}	S_{DIn}	S_{PC}		
`MOV B, SP`														
`JMP B`														
`JMP (B)`														

1. `MOV B, SP`. Guarda en `B` el valor de `SP`.

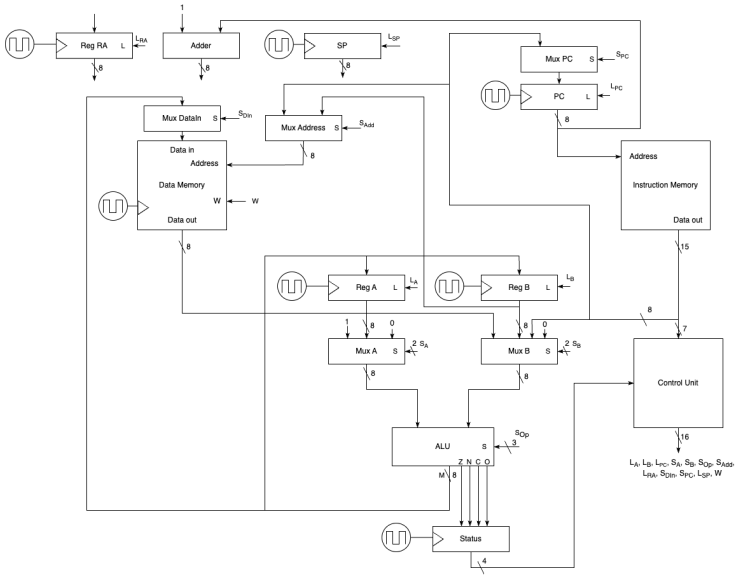
2. `JMP B`. Salta a la instrucción con dirección igual al valor `B`.

3. `JMP (B)`. Salta a la instrucción con dirección igual al valor `Mem[B]`.
4. 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: Computador básico (P2-I2-2024-2)

En arquitecturas RISC, las direcciones de retorno de una subrutina se almacenan en un **registro de enlace** (`RA`), a diferencia del computador básico donde se guardan en el *stack*. Asimismo, estas arquitecturas no poseen `PUSH`/`POP`, sino que acceden al *stack* directamente con el *stack pointer* (`SP`). Deberá contestar las preguntas de esta sección a partir de este contexto y el siguiente diagrama:



(a) **(5 ptos.)** A partir del diagrama de base adjunto, y asumiendo que se eliminan las instrucciones `CALL`, `RET`, `PUSH` y `POP`, realice las modificaciones de *hardware* necesarias e indique la combinación de señales completa para ejecutar las siguientes instrucciones en un ciclo:

- **(0.5 ptos.)** `MOV A,(SP)`. Guarda en `A` el valor `Mem[SP]`.
- **(0.5 ptos.)** `MOV (SP), A`. Guarda en `Mem[SP]` el valor `A`.
- **(1 pto.)** `ADD SP,Lit`. Guarda en `SP` el valor `SP + Lit`. `Lit` puede ser negativo.
- **(1 pto.)** `MOV (SP), RA`. Guarda en `Mem[SP]` el valor `RA`.
- **(1 pto.)** `JAL RA, label`. Guarda `PC+1` en `RA` y salta a la dirección asociada a `label`.
- **(1 pto.)** `JALR RA`. Salta a la dirección almacenada en `RA`.

Instrucción	L_A	L_B	L_{PC}	W	Inc_{SP}	Dec_{SP}	S_A	S_B	S_{OP}	S_{Add}	S_{DIn}	S_{PC}		
`MOV A,(SP)`														
`MOV (SP), A`														
`ADD SP, Lit`														
`MOV (SP), RA`														
`JAL RA, label`														
`JALR RA`														

(b) **(1 pto.)** Modifique el siguiente fragmento de código para que haga uso exclusivo de las instrucciones anteriores. Debe mantener su funcionamiento original y terminar sin errores.

```
MOV A,1
PUSH A
CALL func
POP A
JMP end
func:
  SHL A,A
  CMP A,8
  JEQ end_func
  CALL func
  end_func:
    RET
end:
```

Pregunta 3: Computador básico (P4-I1-2023-2)

En varias arquitecturas se hace uso del registro *base pointer* (`BP`). Este funciona como punto de referencia para obtener los parámetros y dirección de retorno de cada llamada de una subrutina, lo que facilita el manejo de llamados anidados. Deberá modificar la microarquitectura del computador básico para implementar el registro `BP` de 8 bits y su funcionamiento. Se le listarán las instrucciones que deben ser implementadas. Para cada una de ellas, debe incluir la combinación **completa** de señales que las ejecutan. En las señales de carga/escritura/incremento/decremento, debe indicar si se activan (1) o no (0); en las señales de selección, debe indicar el **nombre** de la entrada escogida ("-" si no afecta). Puede realizar todas las modificaciones en un solo diagrama.

(a) (1 pto.) `MOV BP,SP`. Almacena en el registro `BP` el valor del registro `SP`.

(b) (1 pto.) `PUSH BP`. Almacena en `Mem[SP]` el valor de `BP` y decrementa `SP` en una unidad.

(c) (1 pto.) `POP BP`. Incrementa `SP` en una unidad y almacena en `BP` el valor `Mem[SP]`.

(d) (2 ptos.) `MOV Reg,(BP + Lit)`. Almacena en `Reg` (`A` o `B`) el valor `Mem[BP + Lit]`.

Instrucción	L_A	L_B	L_{PC}	W	Inc_{SP}	Dec_{SP}	S_A	S_B	S_{OP}	S_{Add}	S_{DIn}	S_{PC}	L_{BP}		
`MOV BP,SP`															
`PUSH BP`															
`POP BP`															
`MOV A,(BP + Lit)`															
`MOV B,(BP + Lit)`															

(e) (3 ptos.) Asuma que se agregan las instrucciones anteriores a la ISA del computador básico y que se ejecuta el programa adjunto. Explique, a grandes rasgos, lo que realiza la subrutina `func` y señale el valor final de la variable `result`. Puede asumir que `BP` parte en 255.

```
DATA:
  n      3
  result 0
CODE:
  MOV A,(n)          // Dirección Mem. Instr.: 0x00
  PUSH A             // Dirección Mem. Instr.: 0x01
  CALL func          // Dirección Mem. Instr.: 0x02
  POP A              // Dirección Mem. Instr.: 0x03-0x04
  JMP end            // Dirección Mem. Instr.: 0x05
  func:
    PUSH BP          // Dirección Mem. Instr.: 0x06
    MOV BP,SP        // Dirección Mem. Instr.: 0x07
    MOV A,(BP + 3)   // Dirección Mem. Instr.: 0x08
    CMP A,0          // Dirección Mem. Instr.: 0x09
    JEQ base_end_0   // Dirección Mem. Instr.: 0x0A
    CMP A,1          // Dirección Mem. Instr.: 0x0B
    JEQ base_end_1   // Dirección Mem. Instr.: 0x0C
    SUB A,1          // Dirección Mem. Instr.: 0x0D
    PUSH A           // Dirección Mem. Instr.: 0x0E
    CALL func        // Dirección Mem. Instr.: 0x0F
    POP A            // Dirección Mem. Instr.: 0x10-0x11
    MOV A,(BP + 3)   // Dirección Mem. Instr.: 0x12
    SUB A,2          // Dirección Mem. Instr.: 0x13
    PUSH A           // Dirección Mem. Instr.: 0x14
    CALL func        // Dirección Mem. Instr.: 0x15
    POP A            // Dirección Mem. Instr.: 0x16-0x17
    JMP end_func     // Dirección Mem. Instr.: 0x18
    base_end_1:
      INC (result)   // Dirección Mem. Instr.: 0x19
    base_end_0:
      NOP            // Dirección Mem. Instr.: 0x1A
    end_func:
      POP BP         // Dirección Mem. Instr.: 0x1B-0x1C
      RET            // Dirección Mem. Instr.: 0x1D-0x1E
  end:
```

Pregunta 4: Assembly con subrutinas (P3-I1-2024-1)

(a) Indique los valores de los registros `A` y `B` al finalizar la ejecución del código **(a)**.

(b) Indique los valores de los registros `A` y `B` al finalizar la ejecución del código **(b)**.

(c) El código **(c)** *debería* computar la multiplicación entre las variables `X` e `Y`, pero cuenta con errores que generan un resultado erróneo. Indique el valor de los registros `A`, `B` y la variable `res` al finalizar la ejecución del código. Luego, señale dónde se encuentra el o los errores y cómo se resuelven.

(a)

```
DATA:
    res     0

CODE:
    JMP _start

    func_1:
        SHR A, A
        JCR func_2
        CMP A, 0
        JNE func_1
        RET

    func_2:
        NOT B, A
        MOV (B), 14
        INC (res)
        JMP func_1

    _start:
        MOV A, 6
        CALL func_1
        MOV B, (res)
```

(b)

```
DATA:
    var   10

CODE:
    JMP _start

    func_1:
        MOV B, (var)
        NOT A, A
        ADD A, 1
        ADD B, A
        RET

    _start:
        MOV A, 3
        CALL func_1

    end:
        MOV A, (255)
```

(c)

```
DATA:
    res     0
    X       3
    Y       4
    iter    0

CODE:
    MOV A, (Y)
    PUSH A
    CALL mult
    MOV A, (res)
    JMP end

    mult:
        POP B
    sumar:
        MOV A, (res)
        ADD A, (X)
        MOV (res), A
        INC (iter)
        MOV A, (iter)
        CMP A, B
        JNE sumar
        RET

    end:
```

Consideraciones:

1. La primera instrucción del segmento de `CODE` se guardará en la dirección 0 de la memoria de instrucciones.
2. En las instrucciones de *shifting*, el bit descartado se almacena en el *condition code* de *carry*.

Pregunta 5: Assembly con subrutinas (P1-I1-2023-2)

Indique el valor final de los registros `A` y `B` luego de ejecutar el siguiente fragmento de código escrito en el Assembly del computador básico.

```
CODE:
    MOV A,7
    MOV B,1
    PUSH B
    PUSH A
    RET
    MOV A,0
    MOV B,0
    JMP end
    end:
```