Práctico 1
Introducción al lenguaje ensamblador Z80

Ejercicio 1

A = 4Ah = 0100 1010b B = 8Ch = 1000 1100b F = 00X0 X0X1b

a) LD B,24h → A = 0100 1010b B = 0010 0100b F = 00X0 X0X1b

b) AND B → A = 0000 1000b B = 1000 1100b F = 00X1 X000b

c) RLC A → A = 1001 0100b B = 1000 1100b F = 10X0 X000b
Notar que la instricción RLC A no es igual que RLCA, difieren en como alteran las banderas.

d) INC A → A = 0100 1011b B = 1000 1100b F = 00X0 X001b
Tener en cuenta que INC no modifica la bandera de Carry.

Ejercicio 2

A = A7h B = 26h C = FDh F = 1100 0000b

Solo se indican los registros alterados.

a) OR 32h → A = (1010 0111b or 0011 0010b) = 1011 0111b = B7h
F = 10X0 X100b

b) LD A,B → A = 26h

c) INC C → C = FEh F =10X0 X000b

d) CP B → A - B =
```
 1010 0111b
-0010 0110b        → F = 10X0 X010b
-------------------
 1000 0001b
```

No tengo “borrow” del bit 8 al 7 ni del bit 4 al 3 → C = 0 y H = 0 respectivamente.
Notar que los registros A y B no se alteran, solo se afecta el registro F.
Esta instrucción se utiliza para afectar solo el registro F y luego realizar una bifurcación condicional.

e) RLA → A = 0100 1110b F = 11X0 X001b
Notar que la instricción RLA no es igual que RL A, difieren en como alteran las banderas.

Ejercicio 3

a) Testeo A_0
Si $A_0 = 0$ entonces resteo $(8000h)_4$
sino seteo $(8000h)_4$

```
        LD      HL, 8000h
        BIT     0,A             ; Si el A0 “NO ES CERO” → UNO
        JR      NZ,UNO          ;                 sino → CERO

CERO:   RES     4,(HL)          ; (8000h)4 ← 0
        JR      FIN
UNO:    SET     4,(HL)          ; (8000h)4 ← 1
FIN:
```

b) A ← A and 0000 0001b
Si A = 0 entonces (8000h) ← (8000h) and 1110 1111b
sino (8000h) ← (8000h) or 0001 0000b

```
        AND     0000 0001b      ; Modifica banderas. Si A0 = 0 → Z = 1; sino → Z = 0
        LD      A,( 8000h)      ; No modifica banderas
        JR      Z, CERO         ; Si Z=1→CERO; sino → UNO

UNO:    OR      0001 0000b      ; (8000h)4=1
        JR      FIN

CERO:   AND     1110 1111b      ; (8000h)4=0

FIN:    LD      (8000h),A
```

Ejercicio 4

Inicio contador ← 8 (para contar el número de bits que voy a comparar)
Inicializo registro ← 0 (para almacenar número de unos)
Leo la dirección FFh de entrada
Repetir
Roto a la izq y desbordo en el carry.
registro ← registro + 0 + CY
contador ← Decremento (contador)
hasta que contador = 0
Copio registro de almacenamiento en variable CUANTOS

```
PUERTO  EQU     FFh             ; Define la constate PUERTO con el valor FFh
        ORG     8000h           ; Indica que le siguiente línea comienza en la dirección 8000h
CUANTOS DB                      ; Reserva el byte en momoria en la dirección 8000h

; DB , EQU y ORG son directivas para el compilador, no son instrucciones.

        ORG     100h            ; El código comienza a partid de la dirección 100h
        LD      B,8h            ; Inicializo contador
        IN      A,(PUERTO)      ; Leo la dirección FFh de entrada
        LD      E,A             ; Guardo el dato leido en otro registro porque voy a utilizar A
        LD      A,0h            ; Inicializo regisro de almacenamiento

LOOP:   RLC     E               ; Roto izq con carry (E7 → CY)
        ADC     A,0             ; A = A + 0 + CY
        DJNZ , LOOP             ; B ← B - 1
                                ; si B <> 0 → LOOP
                                ; sino continúo con el programa.
        LD (CUANTOS),A          ; aquí A tendrá el valor que necesito, entonces lo guardo.
```

Si no se comprende la función de las directivas DB y ORG, no se preocupe, lo comprenderá más adelante en el curso. Lo importante es que se tenga en cuenta que NO SON INTRUCCIONES.

Ejercicio 5

Primero se debe analizar como saber si un número (dependiendo de cada caso) es mayor que otro.

i) Enteros positivos (binarios sin signo):

- Si A ≥ B ➔ A - B ≥ 0 ➔ no hay “borrow” al hacer A – B ➔ Cy = 0
- Si A < B ➔ A - B < 0 ➔ si hay “borrow” al hacer A – B ➔ Cy = 1

Entonces la idea es hacer la diferencia y utilizar la bandera de carry para decidir.

```
        CP    B          ; A – B , solo se afectan las banderas, registro F.
        JP    C,ELSE
THEN:   LD    C,4h       ; Si Cy = 0 ➔ C = 4h
        JR    FIN
ELSE:   LD C,5h          ; Si Cy =1 ➔ C = 5h
FIN:    .....
```

Un error muy común es utilizar la bandera de signo, que no es correcto, pues esta bandera tiene sentido si el número a considerar es en complemento a 2, que no es el caso.

ii) Números en 2C

- Si $A \geq B$ ➔ $A - B \geq 0$ ➔ al hacer A – B, OVERFLOW = SIGNO
- Si $A < B$ ➔ $A - B < 0$ ➔ al hacer A – B, OVERFLOW = !SIGNO

En resumen se debe hacer:

- A – B
- si OV = 0 entonces
  - si S = 0 → $A \geq B$
  - si S = 1 → $A < B$
- si OV = 1 entonces
  - si S = 0 → $A < B$
  - si S = 1 → $A \geq B$

```
        CP  B              ; Modifico solo las banderas con A - B
        JP  PE, OVERFLOW   ; si OV=1 voy a  OVERFLOW
        JP  P, THEN        ; OV = 0 y S = 0 voy a THEN (que implca A ≥ B)
        JP  ELSE           ; OV = 0 y S = 1 voy a ELSE (que implica A < B)
OVERFLOW: JP  N,THEN       ; idem para OV = 1
ELSE:   LD  C,5h
        JR  FIN
THEN:   LD  C,4h
FIN:    -----
```

iii) Números en magnitud y signo.

Aquí, las banderas disponibles no brindan información suficiente para poder deducir el resultado a partir de la resta (A – B), por lo tanto lo que se debe hacer es discutir según el signo de A y de B.

Veamos:

- Si $A \geq 0$ y $B < 0$ entonces $A > B$
- Si $A < 0$ y $B \geq 0$ entonces $A < B$
- Si $A \geq 0$ y $B \geq 0$ entonces (se reduce a aplicar el caso de enteros positivos)
  - si al hacer A – B, Cy = 0 entonces $A \geq B$
  - sino $A < B$
- Si $A < 0$ y $B < 0$ entonces (es similar, pero con Cy = 1 y analizando el caso A = B).
  - si al hacer A – B, Cy = 1 entonces $A > B$
  - sino si Z = 1 entonces A = B (por lo que $A \geq B$)
  - en otro caso, $A < B$

```
        BIT   7,A          ; Me fijo el signo de A
        JR    Z, APOS      ; si A ≥ 0 voy a APOS
ANEG:   BIT   7,B          ; Me fijo el signo de B
        JR    Z, ELSE      ; A < 0 y si B ≥ 0 voy a ELSE
        CP    B            ; Tengo A < 0 y B < 0,  aplico A - B
        JP    C,THEN       ; si Cy = 1 entonces A > B , voy a THEN
```

```
            JP      Z, THEN     ; si Z = 1, entonces A = B, voy a THEN
            JP      ELSE        ; en otro caso ELSE
APOS:       BIT     7,B         ; Me fijo el signo de B
            JR      NZ, THEN    ; si A ≥ B y B < 0, entonces voy a THEN
            JP      C,ELSE      ; Esto es igual que enteros positivos.
THEN:       LD      C,4h
            JP      FIN
ELSE:       LD      C,5h
FIN:        ----
```

CONCLUSIÓN: Podemos ver que el código a implementar depende del típo de datos que estamos manejando. El Z80 no sabe cual es este tipo de dato, siempre va a dar todas las banderas disponibles, es el programadaor quien debe utilizar estas banderas en forma adecuada.

Ejercicio 6

a) Divido A entre 2^B (B entero positivo).

i) A entero positivo

```
            INC     B
            DEC     B
            JP Z    FIN         ; si B = 0 --> No divido nada y me voy a FIN
DIVIDE:     SRL     A           ; 0, A[7..1] ➔ A y A[0] ➔ Cy. Corrimiento a la derecha con carry.
                                ; Es lo mismo que dividir por 2 con redondeo hacia 0.
            DJNZ    DIVIDE      ; B ← B-1 , si B <> 0 voy a DIVIDE, sino termino
FIN:        ----
```

ii) A número en 2C

```
            INC     B
            DEC     B
            JP Z    FIN         ; hasta acá es igual que antes.

DIVIDE:     SRA     A           ; A[7], A[7..1] ➔ A y A[0] ➔ Cy. Corrimiento a la derecha con
                                ; carry y mantengo el signo.
                                ; Divide entre 2 con redondeo hacia menos infinito.
            JP      P,SIGO      ; (*) Si A es positivo, no incremento.
            JP      NC,SIGO     ; (*) Si A es par, tampoco incremento.
            INC     A           ; (*) Si A < 0  e impar, entonces le sumo 1 al resultado.
SIGO:       DJNZ ,  DIVIDE      ; B ← B-1 , si B <> 0 >➔ DIVIDE
FIN:        ----
```

(*) La instrucción SRA divide un número en complemento a 2 entre 2 y redondea el resultado hacia el menor entero (hacia menos infinito). Al agregar los JP's y el INC, realizo el redondeo hacia 0, independientemente de que el valor incial sea positivo o negativo.
Si no se pusiera esto, nunca daría 0 la división de un número negativo (aunque N > 8).

b) Multiplico A por 2^B (B entero positivo).

i) A entero positivo

```
            INC     B
            DEC     B
            JP Z    FIN         ;si B = 0 no se modifica el resultado, me voy a FIN.

MULT:       SLA     A           ; A ← A[6..0],0 y Cy ← A[7]. Corrimiento a la izquierda.
            JR      C, ERROR    ; Si CY = 1 entonces hubo desbordamiento, voy a ERROR.
            DJNZ    MULT        ; B = B - 1 , si B <> 0 entonces voy a MULT.
```

```
OK:     LD    C,0FFh     ; El resltado queda en A y es correcto. Cargo C ← FFh.
        JR    FIN
ERROR:  LD    C,0h       ; El resultado no es correcto, C ← 00h y A no importa.
FIN     ----
```

ii) A número en 2C

```
        INC   B
        DEC   B
        JP Z  FIN        ;si B = 0 entonces no divido nada y me voy a FIN

MULT:   ADD   A          ; A ← A+A (pues ahora necesito modificar la bandera OV)
        JP    PE, ERROR  ; Si V = 1 entonces hay overflow y voy a ERROR
        DJNZ  MULT       ; B = B - 1  , si B <> 0 entonces voy a MULT.
OK:     LD    C,0FFh     ; El resltado queda en A y es correcto. Cargo C ← FFh..
        JR    FIN
ERROR:  LD    C,0h       ; El resultado no es correcto, C ← 00h y A no importa.
FIN:    ----
```