4. – Repertorio de Instrucciones

Introducción a los microprocesadores
2015

Repertorio de Instrucciones

- Son en total 158 instrucciones.
- Compatibilidad “hacia atrás” con el uP 8080 de Intel.
- Pueden tener entre 1 y 4 bytes.
  - Las instrucciones más largas son las más lentas.
  - 1, 2 y en algunos casos 3 bytes de OPCODE
  - 0, 1 o 2 bytes de “parámetros” (direccionamiento).

Repertorio de Instrucciones

- Cartilla:

(Ejemplo pag. 9.)

Cantidad de períodos de reloj que dura la ejecución.

Cantidad de ciclos de máquina necesarios para ejecutar la instrucción

Mnemonic	Symbólic	Flags S Z - H - P/V N C	OpCode 76 543 210	N° Bytes	N°M Cyc	N° T
LD r, n	r ← n	· · X · X · · ·	00 r 110 ← n →	2	2	7

Comments	
r , r'	Reg
000	B
001	C
010	D
011	E
100	H
101	L
111	A

Cantidad de bytes que ocupa en memoria

Código de operación + los "parámetros"
Ej: LD H,0x25 ➔Opcode: 00 **100** 110.
➔ Parámetro: 0010 0101

Repertorio de Instrucciones

- Grupos de instrucciones:

1. Transferencia de 8 bits
2. Transferencia de 16 bits
3. Intercambio, transferencia de bloque, búsqueda
4. Lógicas y aritméticas de 8 bits
5. Aritméticas de propósito general y control de CPU
6. Aritméticas de 16 bits
7. Rotacion y despñazamiento
8. Bit Set, Reset y Test
9. Transferencia de control
10. Entrada y Salida

La descripción de las instrucciones a continuación no pretende ser exhaustiva.

Instrucciones de Transferencia

- Las transferencias pueden ser:
  - De 8 bits Ej. LD A,B A ← B
  - De 16 bits Ej. LD HL, (nn) H ← (nn+1), L ← (nn)

- Nunca una transferencia es de memoria a memoria.
- El operando origen NO se altera.

Instrucciones de Transferencia

¿Como puedo transferir datos de memoria a memoria?

Ej.1

Transferir1 byte desde la dirección “origen” a la dirección “destino”.

```
LD A, (origen);Indirecto
LD (destino), A
```

Ej. 2

Transferir 2 bytes usando solo transferencias de 8 bits.

```
LD HL, origen ;inmediato
LD DE, destino
LD A, (HL)
LD (DE), A
INC HL
INC DE
LD A, (HL)  ; Indirecto
              ; por registro
LD (DE), A
```

Instrucciones de Transferencia

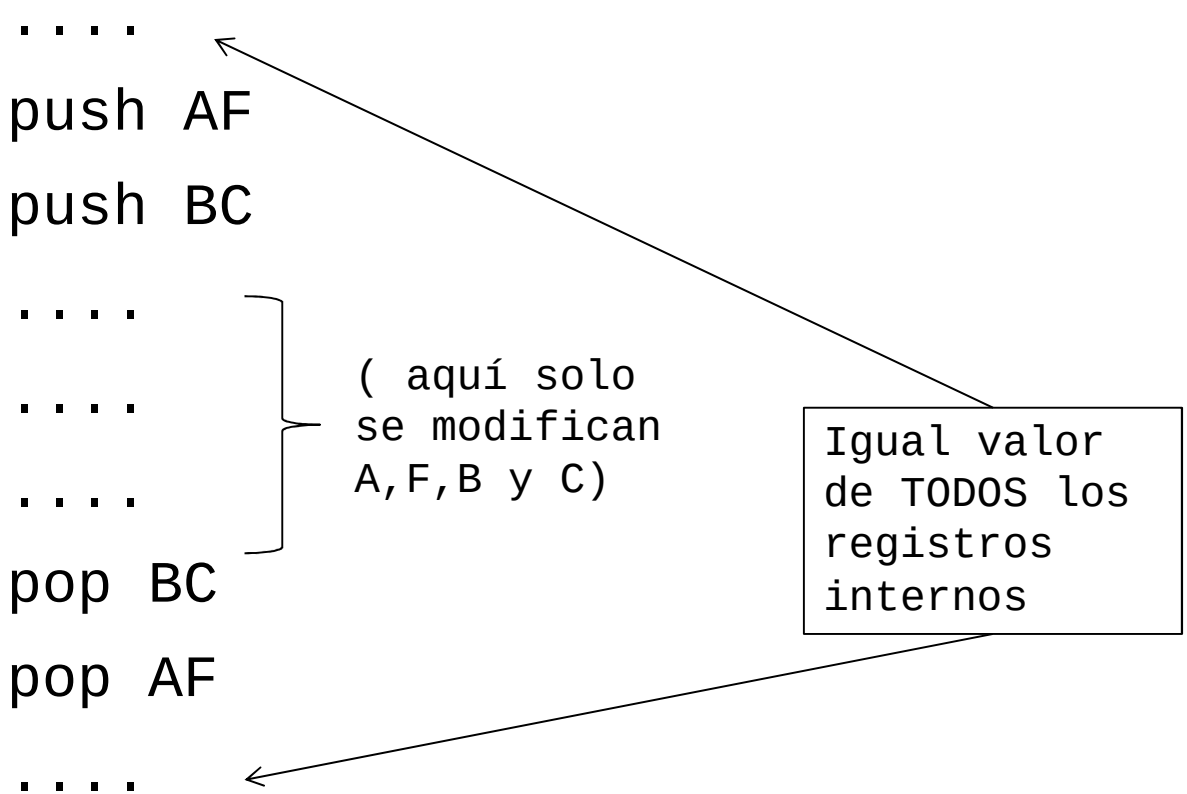
- Stack
  - "Crece" hacia abajo.
  - SP apunta al último lleno
  - Transferencias de 16 bits
  - Instrucciones
    - `Push reg16`
    - `Pop reg16`

Pag.	Mnemonic	Symbólic	Flags S Z - H - P/V N C	OpCode	N° Bytes	N° M	N° T
11	PUSH qq	(SP-2)← qq_L (SP-1)← qq_H SP← SP-2	· · X · X · · ·	11 qq0 101	1	3	11
11	POP qq	qq_H ←(SP+1) qq_L ←(SP) SP← SP+2	· · X · X · · ·	11 qq0 001	1	3	10

qq	Par de registros
00	BC
01	DE
10	HL
11	AF

Instrucciones de Transferencia

- Se utiliza para preservar contexto



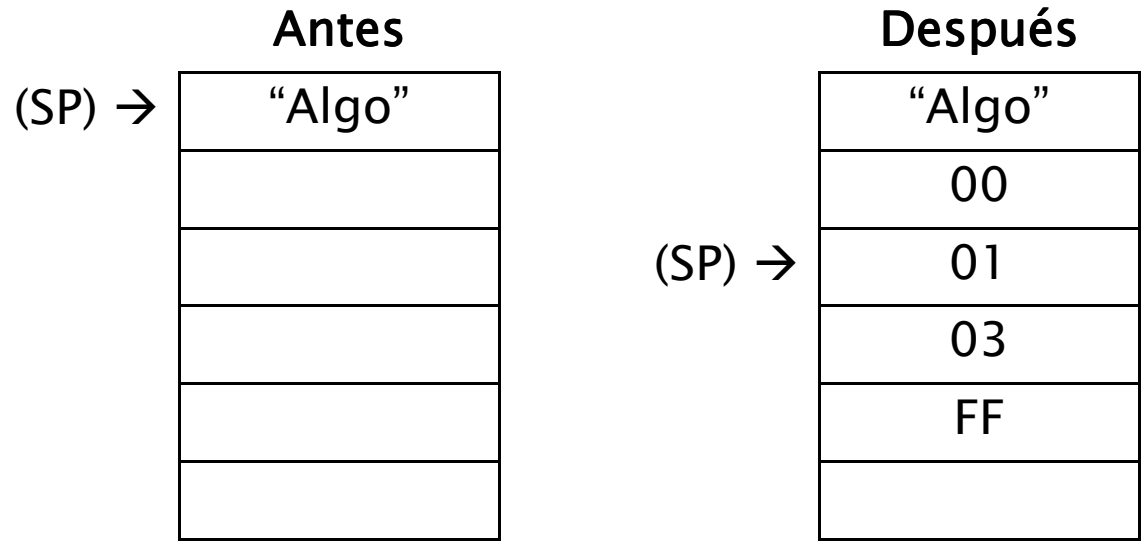
Instrucciones de Transferencia

- Ejemplo: Tengo el programa

```
. . .
PUSH  BC
PUSH  HL
POP  DE
. . .
```

Antes:
BC = 00 01
DE = 0B 07
HL = 03 FF

Después:
BC = 00 01
DE = 03 FF
HL = 03 FF



Instr. Lógicas y Aritméticas de 8 bits

ALU de 8 bits + Acumulador + banderas (S,Z,H,P/V,N,C)

- **Instrucciones**
  - **ADD / ADC**
  - **SUB / SBC**
  - **AND / OR / XOR**
  - **CP**
  - **INC / DEC**

- Comparaciones (CP)
  - CP “operando de 8 bits”
  - Ej: CP B ;A - B
    - Afecta banderas
    - **Descarta el resultado**.
  - Se utiliza para testear condiciones:

```
        . . . .
        CP B
        JP NZ, DIR
        . . . .
DIR:    . . . .
```

Instr. Lógicas y Aritméticas de 8 bits

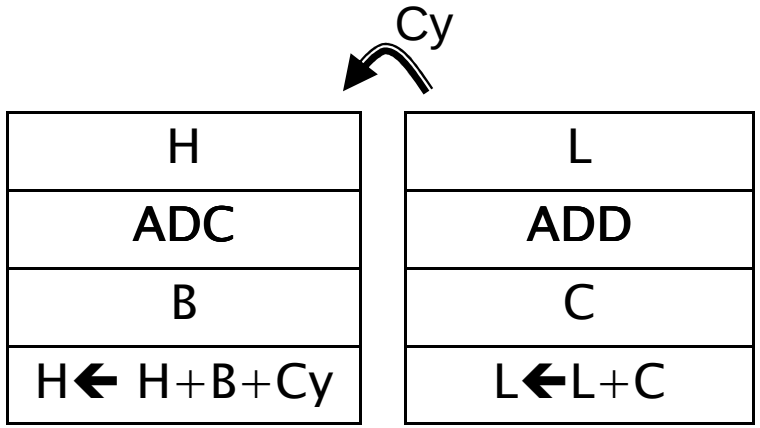
- Ejemplo:

  Utilizando sumas de 8 bits, sumar:

  $$HL \leftarrow HL + BC$$

  1. Suma los menos significativos L + C. Afecta el Carry
  2. Suma los más significativos con el carry: H + B + Cy

```
ld a, l
add c       ; a ← a + c
ld l, a
ld a, h
adc b       ; a ← a + b + Cy
ld h, a
```



Instrucciones Aritméticas de propósito general y control de CPU

- Que operan sobre A
  - `NEG  Negado en complemento a 2`
  - `CPL  Complemento bit a bit.`
- Que operan sobre flag Cy
  - `SCF  Poneen 1 la bandera de C`
  - `CCF  Complementa la bandera de C`
- Relleno, detención
  - `NOP  No hace nada`
  - `HALT  Detiene la CPU (ejecuta NOPs)`
- Ajuste de suma decimal
  - `DAA  (corrige una suma o resta BCD)`
- Control de interrupciones
  - `DI / EI / IM 0 / IM 1 / IM 2`

Instrucciones Aritméticas de 16 bits

- ALU 16 bits: Sumas, restas, incrementos, decrementos
- INC y DEC no afectan flags
  - Trucos, p. ej. para saber si HL es cero

```
ld a, l
or h          ; afecta banderas
```

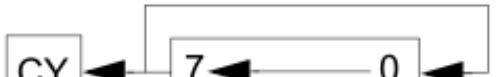
- No hay instrucción que copie SP a otro registro
  - Trucos:

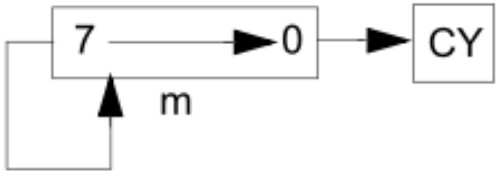
```
1) ld (dir), sp
   ld hl, (dir)
```

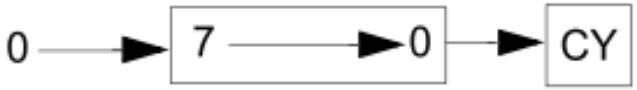
```
2) ld hl, 0
   add hl, sp
```

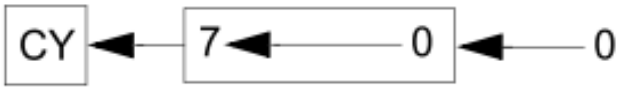
Instr. de Rotación y Desplazamiento

- RLC (Rotate Left Circular)
- RL (Rotate Left)
- SRA (Shift Right Arithmetic)
  - División entera por 2
- SRL (Shift Right Logic)
- SLA (Shift Left Arithmetic)
- Ejemplo: multiplicar por 10
  - A ← B * 10
  - A ← (B * 8) + (B * 2)










```
SLA  B          ; x 2
LD   A, B
SLA  B          ; x 4
SLA  B          ; x 8
ADD  B
```

Instr. de Saltos, Llamadas y Retornos

- Modifican el PC
- Salto (Jump)
  - Incondicional: JP dir
  - Condicional: JP Z, dir
    - En el ejemplo salta si fue 0 el resultado de la última operación (si la bandera Z está “prendida”)
- Salto
  - Absoluto (JP): PC ← dir
    - ocupa 3 bytes (opcode, dirL, dirH)
  - Relativo (JR): PC ← PC + desp8 (en compl. a 2)
    - ocupa solo 2 bytes
    - el ensamblador se encarga

Se utilizan para implementar if, case, for, while, etc

Instr. de Saltos, Llamadas y Retornos

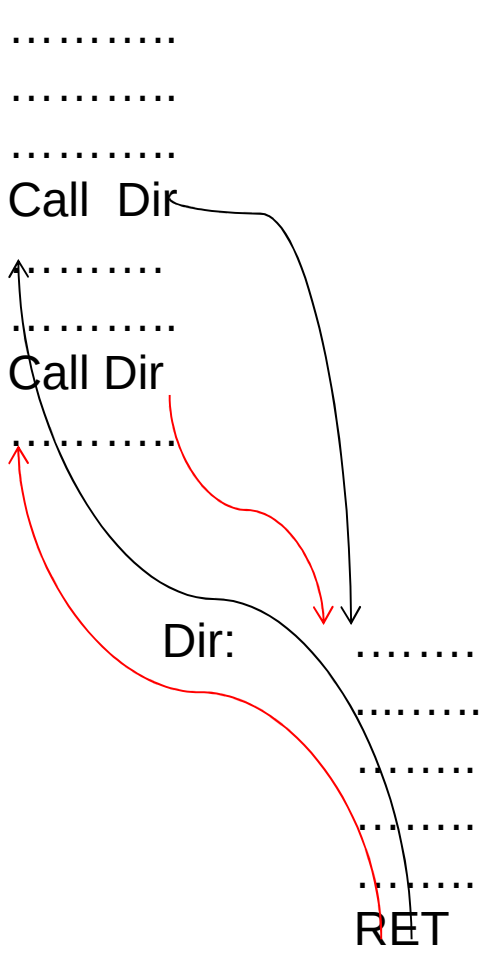
Ejemplo:

- Mover N bytes
  - B: N bytes a mover
  - HL: origen
  - DE: destino

```
        LD B, cant
        LD HL, origen
        LD DE, destino
repetir:
        LD A, (HL)
        LD (DE), A
        INC HL
        INC DE
        DEC B      ;afecta banderas
        JR NZ, repetir
```

Instr. de Saltos, Llamadas y Retornos

- CALL dir / CALL cc,dir
  - Permite llamar al mismo trozo de código desde diferentes partes de un programa.
- RET
  - Retorna a la instrucción siguiente a CALL
- ¿Cómo lo hace?
  - CALL guarda dirección de retorno en el stack.
    - Sería equivalente a:
      - "PUSH PC"
      - JP dir
  - RET toma los datos del stack y lo carga en contador de programa
    - Sería equivalente a:
      - "POP PC"



Instr. de Saltos, Llamadas y Retornos

- Mismo ejemplo como subrutina

- Prog. ppal.:

```
        ...
        LD B, cant1
        LD HL origen1
        LD DE, destino1
        CALL mover
        ....
        LD B, cant2
        LD HL origen2
        LD DE, destino2
        CALL mover
```

```
mover:
        LD A, (HL)
        LD (DE), A
        INC HL
        INC DE
        DJNZ mover
        RET
```

Instrucciones de Entrada/Salida

- Hardware
  - Se activa señal de control /IORQ
  - Valen solamente los 8 bits menos significativos de direcciones (A7..A0)
  - Solo 256 puertos de entrada y 256 de salida

- Instrucciones
  - Direccionamiento directo (solo al acumulador)
    - IN A, (dir8) / OUT (dir8), A
  - Direccionamiento indirecto por registro (con C)
    - IN reg, (C) / OUT (C), reg
  - Transferencias en bloque
    - INI / INIR / IND / INDR
    - OUTI / OUTIR / OUTD / OUTDR

Instrucciones de Acceso de a Bit

- Instrucciones de a BIT
  - BIT b, “operando de 8 bits” Z ← NOT(Bit b del operando)
  - SET b, “operando de 8 bits” Bit b del operando ← 1
  - RES b, “operando de 8 bits” Bit b del operando ← 1

  - Ejemplo de operaciones equivalentes con máscaras
    - SET 4, A es equivalente a OR A, 0001 0000

Instrucciones de Intercambio

- Intercambio de bancos de registros

EX AF, AF‘	AF ↔ AF’
EXX	BC ↔ BC’ DE ↔ DE´ HL ↔ HL´

- Intercambio de otros registros

EX DE, HL	DE ↔ HL
EX (SP), HL	H ↔ (SP+1) L ↔ (SP)
EX (SP), IX	H ↔ (SP+1) L ↔ (SP)
EX (SP), IY	H ↔ (SP+1) L ↔ (SP)

Instrucciones de Búsqueda y Transferencia de Bloques

- Instrucciones
  - `LDI / LDIR` `; Copia e incrementa punteros`
  - `LDD / LDDR` `; Copia e decrementa punteros`
  - `CPI / CPIR` `; Compara e inc punteros`
  - `CPD / CPDR` `; Compara e dec punteros`

- Usa registros
  - `BC: cant bytes`
  - `HL: origen`
  - `DE: destino`

- El ejemplo queda:

```
LD BC, cant
LD HL, origen
LD DE, destino
LDIR          ;(DE) ← (HL)
              ; DE ← DE + 1
              ; HL ← HL + 1
              ; BC ← BC – 1
              ; Repetir hasta BC = 0
```