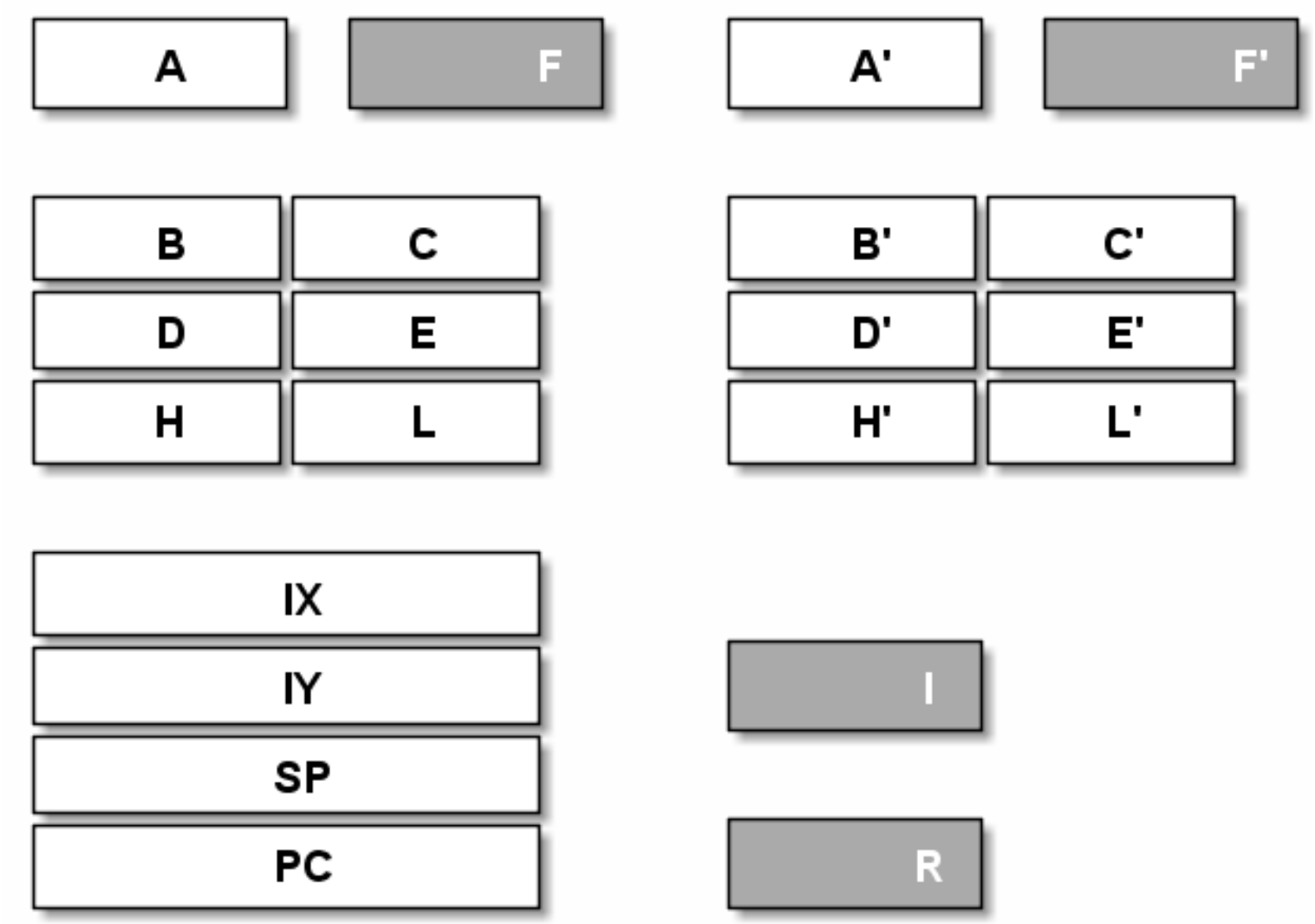
Repaso – Registros internos Z80



Repaso – Formato instrucciones

- Formato con 4 direcciones
- Acumulador: A ← A op oper2
- PC: Program Counter
  - dirección de la próxima instrucción
- Instrucciones de operación

Código de Operación	OP1	~~OP2~~	~~RESULT~~	~~PROX. INSTR.~~

- Instrucciones de salto

Código de Operación	~~OP1~~	~~OP2~~	~~RESULT~~	PROX. INSTR.

Hoja de ruta

- Modos de direccionamiento
- Cartilla de instrucciones
- Grupos de instrucciones

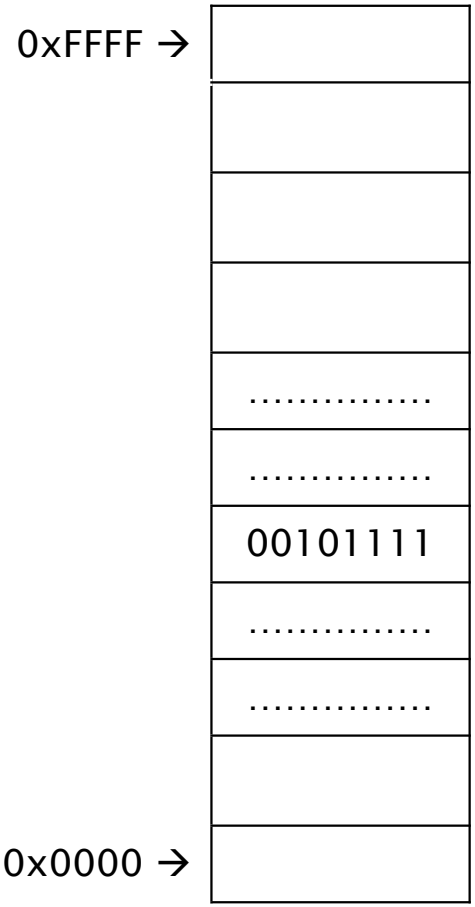
  - Recorrida con ejemplos por el repertorio de instrucciones.
  - No pretende ser exhaustiva.
  - Información detallada: cartilla y manual Z80

Modos de direccionamiento

- Las distintas maneras de especificar en una instrucción de dónde obtener el valor de los operandos, el lugar donde almacenar el resultado o dónde ir a buscar la siguiente instrucción a ejecutar.
  - Implícito
  - Inmediato
  - Registro
  - Directo
  - Indirecto por Registro
  - Indexado
  - De a bit
  - Relativo al programa
  - Página 0

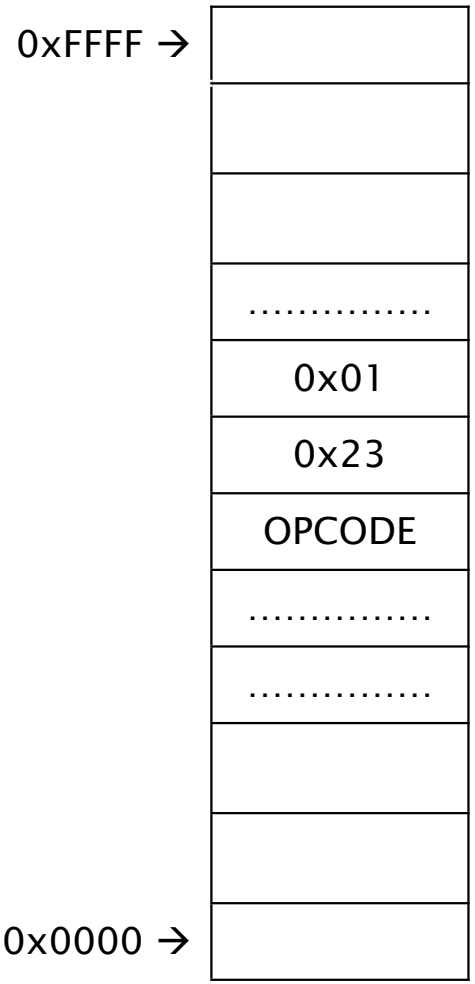
Modos de direccionamiento

- Implícito
  - Implícito en el propio código de operación.
  - Se ejecuta siempre sobre el mismo operando.
  - Ejemplo: CPL
    - A ← not A



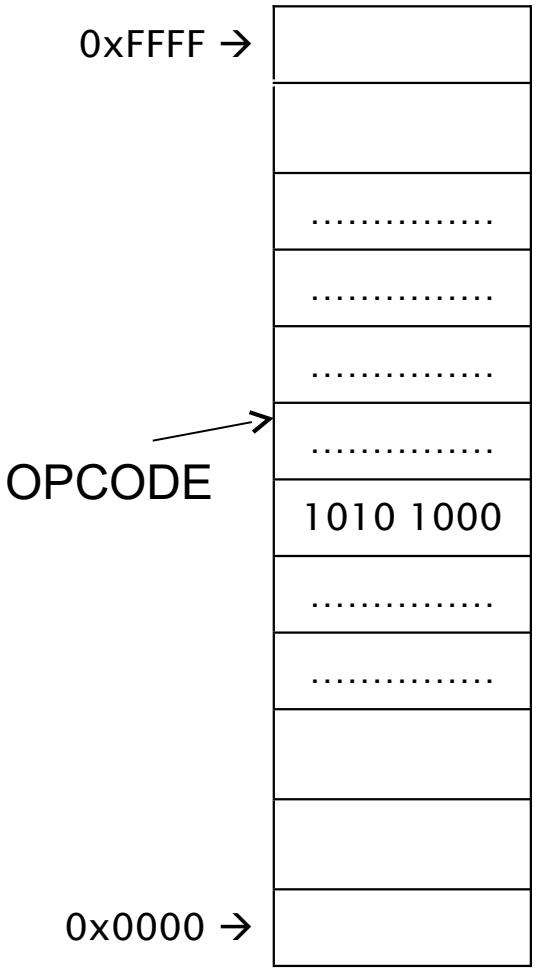
Modos de direccionamiento

- Inmediato (8 y 16 bits)
  - Operando es una cte.
  - A continuación de opcode.
  - ADD A, 0x27
    - A ← A + 0x27
  - LD HL, 0x0123
    - HL ← 0x0123
  - Convención: byte bajo en dirección más baja



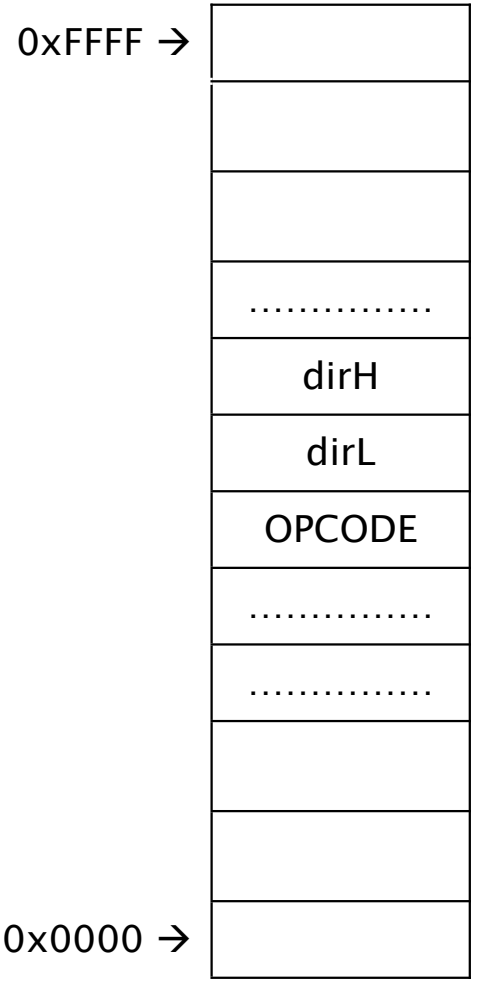
Modos de direccionamiento

- Registro
  - Operando es el contenido de un registro interno.
  - El OPCODE tiene un campo para especificar el registro.
  - OR B
    - A ← A OR B
  - Opcode es 1010 1*rrr*
  - *rrr* = 000 corresponde a B



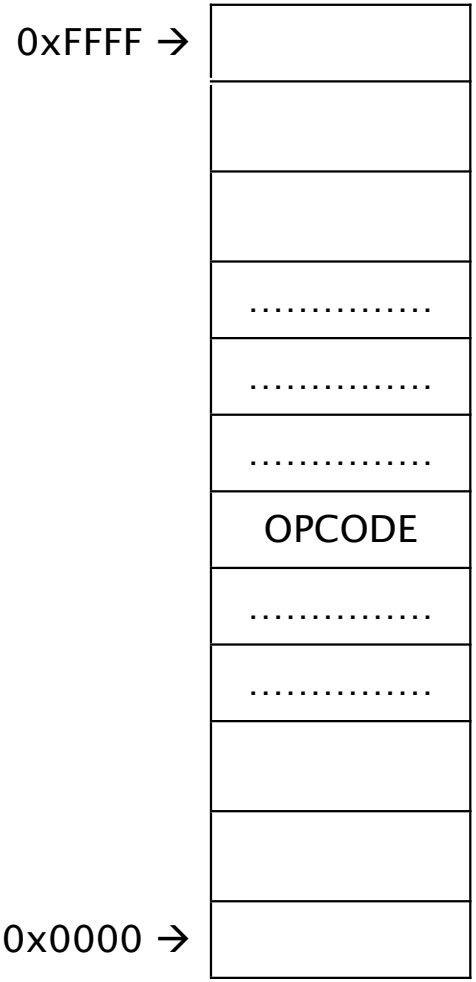
Modos de direccionamiento

- Directo
  - La dirección del operando se suministra a continuación del OPCODE.
  - LD A, (*dir*)
  - *dir* es una dirección de 16 bits
  - (*dir*) indica “el contenido de memoria” en la dirección *dir*
  - p. ej LD A, (0x8000)
    - A ← (0x8000)
  - No confundir con LD A, 0x80



Modos de direccionamiento

- Indirecto por registro
  - La dirección del operando es el contenido de un par de registros.
  - SUB A, (HL)
    - A ← A - (HL)
  - Nuevamente paréntesis indica contenido de memoria
  - Permite calcular dirección del operando.



Modos de direccionamiento

- Indexado
  - La dirección se forma sumando el contenido de uno de los registros índice (IX o IY) con un desplazamiento de 8 bits.
  - Despl en complemento a 2 (entre -127 y 128)
  - Ejemplo: INC (IX+5)
    - Suponiendo IX = 0x9000
    - (0x9005) ← (0x9005) + 1
  - Acceso a campos de una estructura de datos en diferentes lugares de memoria

Modos de direccionamiento

- De a bit
  - Permite acceder a un bit de un registro o posición de memoria
  - Ejemplo: SET 4, A
    - $A_4 \leftarrow 1$
  - Se puede sustituir por operaciones con constantes
    - OR A, 00010000B
    - A la cte 00010000B se le llama *máscara*

Modos de direccionamiento

- Relativo al programa
  - Saltos relativos
  - PC ← PC + desplazamiento
- De Página cero
  - Instrucción Restart
  - PC ← valor prefijado

- Ambos se usan en instrucciones de salto, para especificar dirección de próxima instrucción.
  - En detalle cuando veamos las instrucciones

Resumen – Modos direccionamiento

Implícito	CPL	A ← not A	
Inmediato	ADD A, 0x27	A ← A + 0x27	ctes
Registro	OR A, B	A ← A or B	
Directo	INC (dir)	(dir) ← (dir) + 1	Variables en memoria
Indirecto por Registro	LD A, (HL)	A ← (HL)	Calcular direcciones
Indexado	LD A, (IX+desp8)	A ← (IX+desp8)	
De a bit	SET 5, A	A_5 ← 1	
Relativo al programa	JR	PC ←PC + desp	Salto relativo
Página 0	RST 0x08	PC← 0x0008	Llamada a subrutina en direcc. fija

Hoja de ruta

- Modos de direccionamiento
- Cartilla de instrucciones
- Grupos de instrucciones

Repertorio de instrucciones

- 158 instrucciones
- Compatibilidad *"hacia atrás"* con 8080 de Intel
- 1, 2 y en algunos casos 3 bytes de OPCODE
- Datos adicionales:
  - dato inmediato, dirección de 2 bytes, desplazamiento en direccionamiento relativo.

8-BIT LOAD GROUP

Mnemonic	Symbolic Operation	S	Z	X	H	X	P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD r, r′	r ← r′	•	•	X	•	X	•	•	•	01	r	r′		1	1	4	r, r′ Reg.
LD r, n	r ← n	•	•	X	•	X	•	•	•	00	r	110		2	2	7	000 B
											← n →						001 C
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01	r	110		1	2	7	010 D
LD r, (IX + d)	r ← (IX + d)	•	•	X	•	X	•	•	•	11	011	101	DD	3	5	19	011 E
										01	r	110					100 H
											← d →						101 L
LD r, (IY + d)	r ← (IY + d)	•	•	X	•	X	•	•	•	11	111	101	FD	3	5	19	111 A
										01	r	110					
											← d →						
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01	110	r		1	2	7	
LD (IX + d), r	(IX + d) ← r	•	•	X	•	X	•	•	•	11	011	101	DD	3	5	19	
										01	110	r					
											← d →						
LD (IY + d), r	(IY + d) ← r	•	•	X	•	X	•	•	•	11	111	101	FD	3	5	19	
										01	110	r					
											← d →						
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00	110	110	36	2	3	10	
											← n →						
LD (IX + d), n	(IX + d) ← n	•	•	X	•	X	•	•	•	11	011	101	DD	4	5	19	
										00	110	110	36				
											← d →						
											← n →						

Repertorio de instrucciones

- Cartilla:
  - Mnemonic
  - Operación: descripción similar a RTL.
  - Flags: indicación de cómo afecta a cada bandera.
  - OPCODE: en binario y en hexadecimal.
  - Bytes: cantidad de bytes que ocupa en memoria.
  - Ciclos M: cantidad de ciclos de máquina necesarios para ejecutarla.
  - T States: cantidad de períodos de reloj que dura la ejecución.

Repertorio de instrucciones

- Grupos de instrucciones:
  - Transferencias de 8 bits
  - Transferencias de 16 bits
  - Intercambio, transferencia de bloque, búsqueda
  - Lógicas y aritméticas de 8 bits
  - Aritméticas de propósito general y control de CPU
  - Aritméticas de 16 bits
  - Rotacion y despñazamiento
  - Bit Set, Reset y Test
  - Saltos y llamadas a subrutinas
  - Entrada y Salida

Transferencias

- Instrucción LOAD (mnemonic LD)
- Varios modos de direccionamiento
- De 8 o 16 bits
- Nunca de memoria a memoria
- El operando origen NO se altera

Transferencias

- Copiar 1 byte desde la dirección origen a la dirección destino.

Transferencias

- Copiar 1 byte desde la dirección origen a la dirección destino.

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

- Copiar 2 bytes

Transferencias

- Copiar 1 byte desde la dirección origen a la dirección destino.

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

- Copiar 2 bytes

```
LD HL, origen  ; << Modo?
LD DE, destino
LD A, (HL)
LD (DE), A

INC HL
INC DE
LD A, (HL)  ; << Modo?
LD (DE), A
```

Transferencias - Stack

- Stack
  - “crece” hacia abajo.
  - SP apunta al último ocupado
  - Transferencias 16 bits

- Instrucciones
  - `Push qq`
    - `(SP-1) ← qqH`
    - `(SP-2) ← qqL`
    - `SP ← SP-2`
  - `Pop qq`
    - `qqL ← (SP)`
    - `qqH ← (SP+1)`
    - `SP ← SP+2`
  - `Llamadas y retornos`

Transferencias - Stack

- Uso para para preservar contexto

```
push AF
push BC
...
...
...
pop AF
pop BC
```

modifico AF y BC a gusto

Transferencias - Stack

- Utilización para preservar contexto

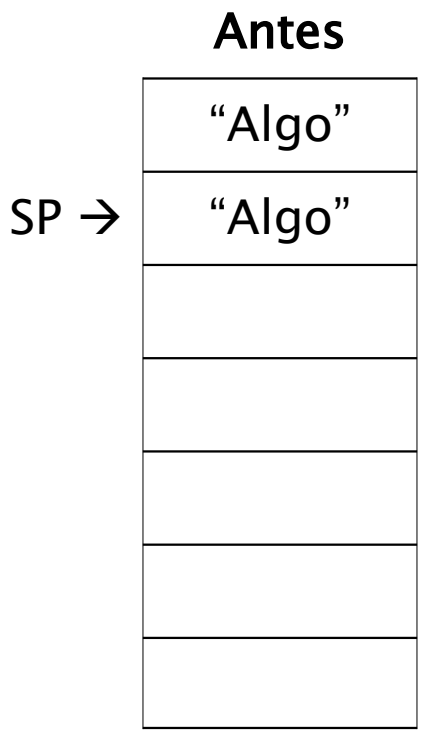
```
push AF
push BC
...
...
...
pop BC
pop AF
```

OJO!!
POPs en orden
inverso a los PUSH

Transferencias - Stack

- Ejemplo. Al inicio:
  - BC = 00 01
  - DE = 0B 07
  - HL = 03 FF

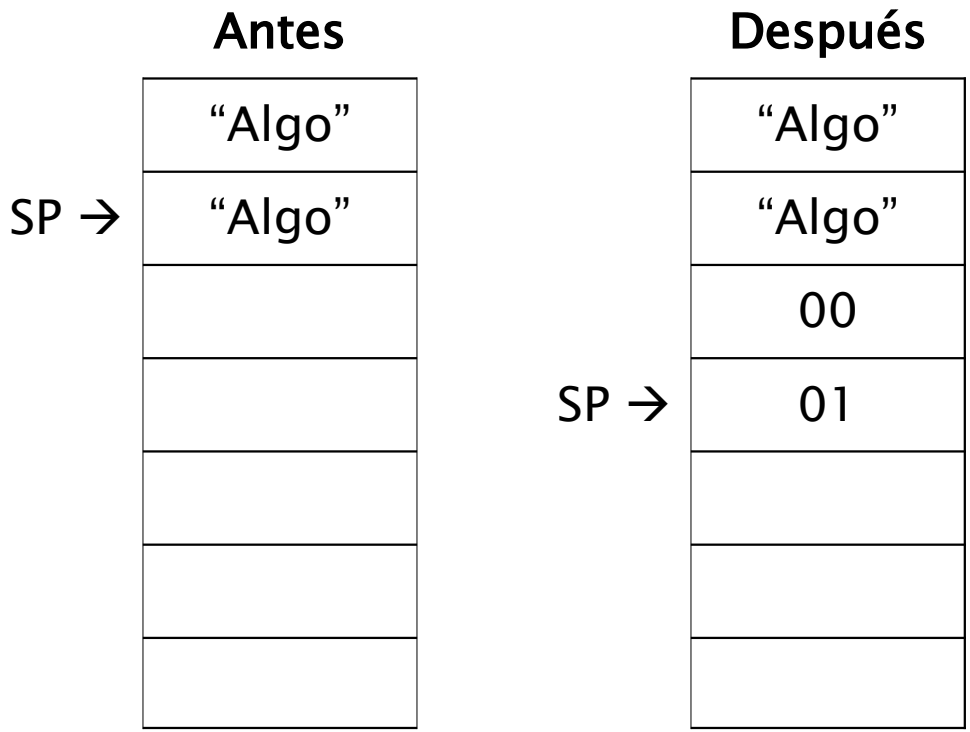
```
...
PUSH BC
PUSH HL
POP DE
...
```



Transferencias - Stack

- Ejemplo. Al inicio:
  - BC = 00 01
  - DE = 0B 07
  - HL = 03 FF

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



Transferencias - Stack

- Ejemplo. Al inicio:
  - BC = 00 01
  - DE = 0B 07
  - HL = 03 FF

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

Transferencias - Stack

- Ejemplo. Al inicio:
  - BC = 00 01
  - DE = 0B 07
  - HL = 03 FF

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

Lógicas y aritméticas de 8 bits

- Acumulador y ALU
- Banderas
  - Acarreo y préstamo
  - Overflow
- Comparaciones (cp)
- Notación cartilla
  - En la suma (add), un renglón para cada modo de direccionamiento
  - Para el resto un solo renglón

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

8-BIT ARITHMETIC AND LOGICAL GROUP

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	10	[000]	r		1	1	4	r Reg.
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	11	[000]	110		2	2	7	000 B
											← n →						001 C
																	010 D
ADD A, (HL)	A ← A + (HL)	↕	↕	X	↕	X	V	0	↕	10	[000]	110		1	2	7	011 E
ADD A, (IX + d)	A ← A + (IX + d)	↕	↕	X	↕	X	V	0	↕	11	011	101	DD	3	5	19	100 H
										10	[000]	110					101 L
											← d →						111 A
ADD A, (IY + d)	A ← A + (IY + d)	↕	↕	X	↕	X	V	0	↕	11	111	101	FD	3	5	19	
										10	[000]	110					
											← d →						
ADC A, s	A ← A + s + CY	↕	↕	X	↕	X	V	0	↕		[001]						s is any of r, n, (HL), (IX + d), (IY + d) as shown for ADD instruction. The indicated bits replace the [000] in the ADD set above.
SUB s	A ← A − s	↕	↕	X	↕	X	V	1	↕		[010]						
SBC A, s	A ← A − s − CY	↕	↕	X	↕	X	V	1	↕		[011]						
AND s	A ← A ∧ s	↕	↕	X	1	X	P	0	0		[100]						
OR s	A ← A ∨ s	↕	↕	X	0	X	P	0	0		[110]						
XOR s	A ← A ⊕ s	↕	↕	X	0	X	P	0	0		[101]						
CP s	A − s	↕	↕	X	↕	X	V	1	↕		[111]						

Lógicas y aritméticas de 8 bits

- Ejemplo:
  - suma de HL con BC (HL ← HL + BC)
  - Suma los menos significativos y afectando flag Carry
  - Suma con acarreo los más significativos

Lógicas y aritméticas de 8 bits

- Ejemplo:
  - suma de HL con BC (HL ← HL + BC)
  - Suma los menos significativos y afectando flag Carry
  - Suma con acarreo los más significativos

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

Aritméticas de propósito general y control de CPU

- Que operan sobre A
  - NEG / CPL
- Que operan sobre flag Cy
  - SCF / CCF
- Relleno, detención
  - NOP / HALT
- Ajuste de suma decimal
  - DAA
- Control de interrupciones
  - DI / EI / IM 0 / IM 1 / IM 2

Aritméticas de 16 bits

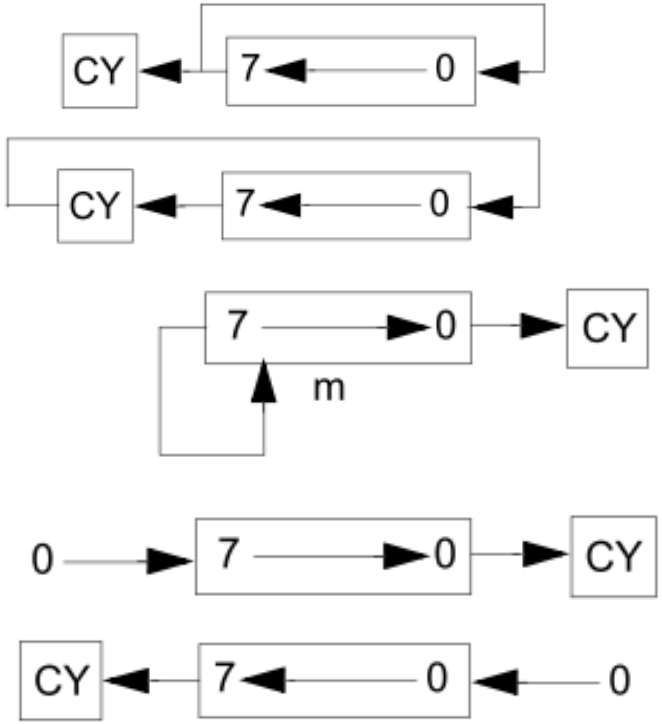
- ALU 16 bits: Sumas, restas, incrementos, decrementos
- INC y DEC no afectan flags
  - Trucos, p. ej. para saber si HL es cero

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

- No hay instrucción que copie SP a otro registro
  - Trucos:
  - O bien: `ld (dir), sp  / ld hl, (dir)`
  - O bien: `ld hl, 0 / add hl, sp`

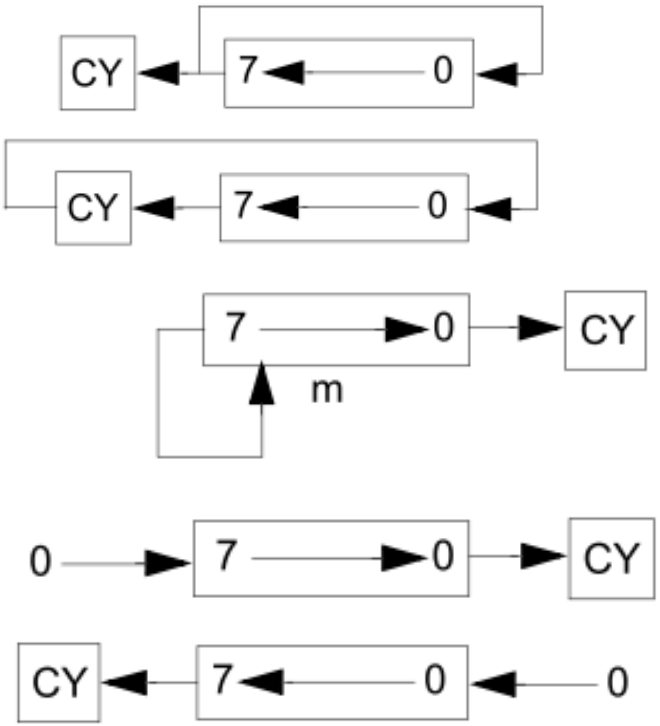
Rotación y desplazamiento

- RLC (Rotate Left Circular)
  - 8 bits, Cy fuera del anillo
- RL (Rotate Left)
  - 9 bits, Cy incluído.
- SRA (Shift Right Arithmetic)
  - División entera por 2
- SRL (Shift Right Logic)
- SLA (Shift Left Arithmetic)
- Ejemplo: multiplicar por 10



Rotación y desplazamiento

- RLC (Rotate Left Circular)
  - 8 bits, Cy fuera del anillo
- RL (Rotate Left)
  - 9 bits, Cy incluído.
- 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
```

Saltos, Llamadas y Retornos

- Modifican el PC
- Salto (Jump)
  - Incondicional: JP *dir*
  - Condicional: JP Z, *dir*
    - En el ejemplo salta si último resultado fue 0 (flag Z está “prendida”)
- Salto
  - Absoluto (JP): PC ← *dir*
    - ocupa 3 bytes (opcode, dirL, dirH)
  - Relativo (JR): PC ← PC + *desp8*
    - ocupa solo 2 bytes
    - Reubicable
    - El ensamblador se encarga de calcular *desp8*
- Utilizados para implementar if, case, for, while, ...

Saltos, Llamadas y Retornos Ejemplo

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

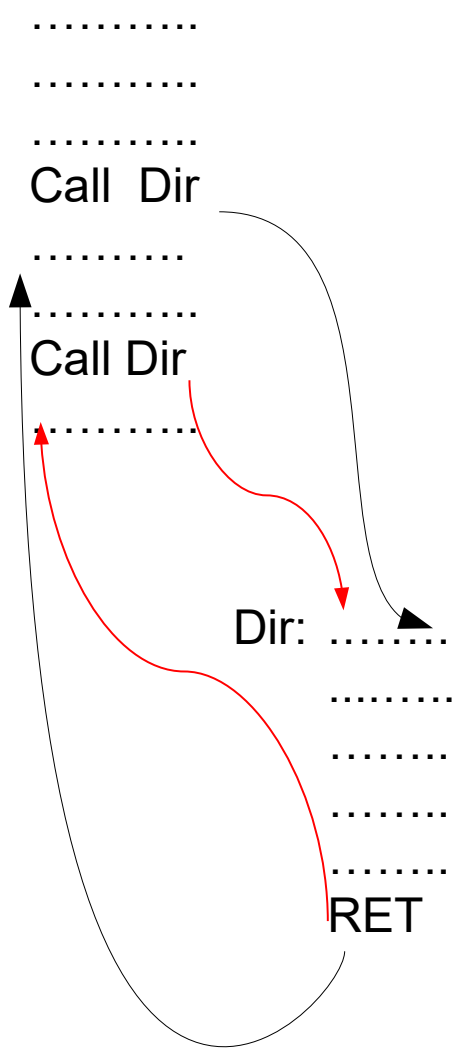
Saltos, Llamadas y Retornos Ejemplo

- 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
    JR NZ, repetir
```

Saltos, Llamadas y Retornos

- CALL *dir*
  - Permite llamar al mismo trozo de código desde diferentes partes de un programa.
- RET
  - Retorna a la instrucción siguiente a CALL
- Mecanismo
  - CALL guarda dirección de retorno en el stack.
    - Sería equivalente a
      - “PUSH PC”
      - JP dir
  - RET toma dato del stack y lo carga en contador de programa
    - Sería equivalente a:
      - “POP PC”



Saltos, Llamadas y Retornos Ejemplo

- 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:
    . . .
    . . .


    . . .
    . . .
    ret
```

Saltos, Llamadas y Retornos Ejemplo

- 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
```

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 con registro C
    - IN reg, (C) / OUT (C), reg
  - Transferencias en bloque
    - INI / INIR / IND / INDR
    - OUTI / OUTIR / OUTD / OUTDR

Acceso de a bit

- Instrucciones de a BIT
  - BIT
  - SET
  - RES

- Operaciones equivalentes con máscaras
  - Ej: RES 7, A equivale a AND A, 0111 1111B

Intercambio

- Intercambio de bancos de registros
  - EX AF, AF'
  - EXX
- Intercambio de contenido de registros
  - EX DE, HL DE ↔ HL
  - EX (SP), HL H ↔ (SP+1)
    L ↔ (SP)
    - Ídem con IX e IY

Transferencias de Bloques

- Uso registros
  - BC: cant bytes
  - HL: origen
  - DE: destino
- Operaciones
  - LDI: Load and increment
  - LDIR: Load, increment and repeat
  - Ídem con decrement
  - Ídem con Compare

- El ejemplo queda:

```
LD BC, cant
LD BC, origen
LD DE, destino
LDIR
```