PRÁCTICAS DE LABORATORIO

Grado en Ingeniería Informática de Gestión y Sistemas de Información
Estructura de Computadores
Pre-laboratorio (ejercicios para casa): ENUNCIADOS

REPASO

En el 8086 se utiliza la **base hexadecimal**.

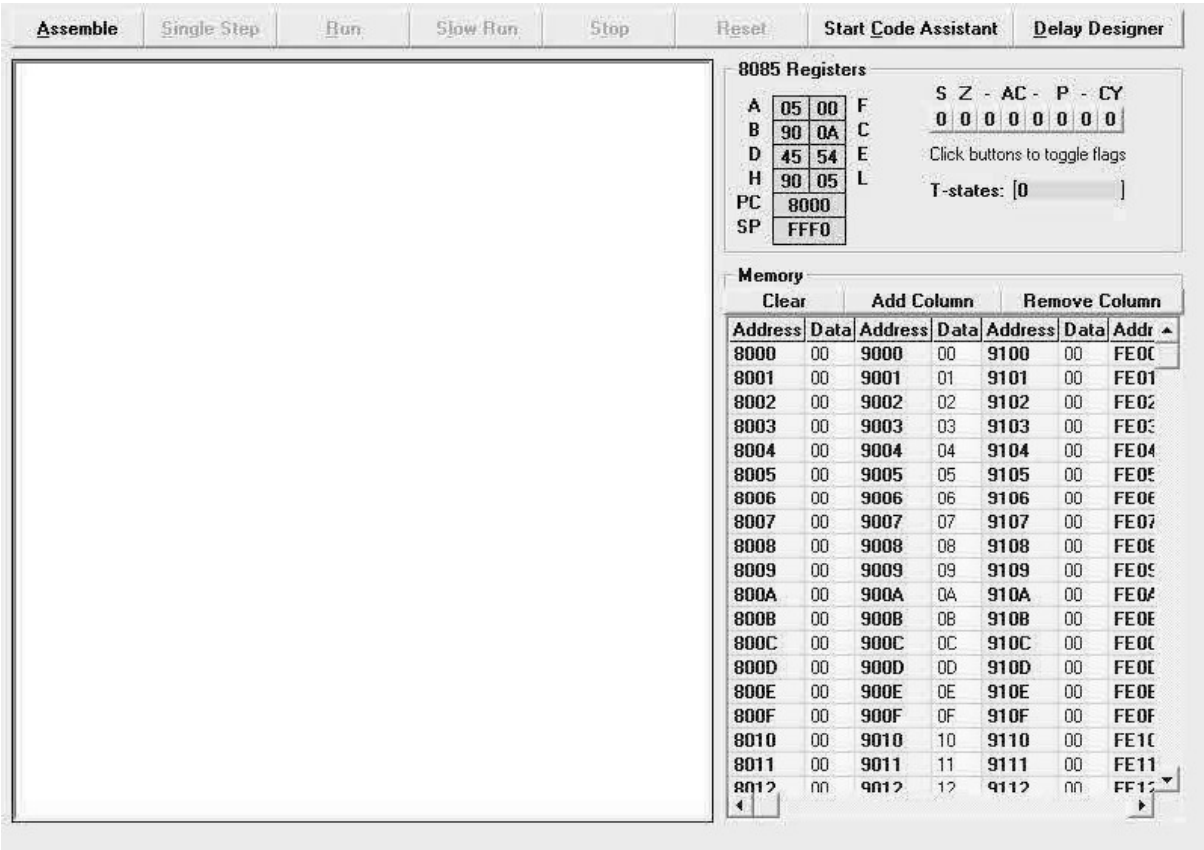
a) ¿Cuáles son los valores que puede tomar un número hexadecimal?

b) ¿Cuántas cifras binarias hacen falta para representar una cifra hexadecimal?

c) ¿Cuántas cifras hexadecimales tienen los registros? ¿Por tanto, cuál es el valor máximo que pueden contener en base 16, en base 2 y en base 10?

d) ¿Cuántas cifras hacen falta para especificar una posición de memoria? Entonces, ¿cuántos registros hacen falta para utilizarlos como apuntador de memoria?

INSTRUCCIONES DE TRANSFERENCIA



1. En la situación del 8086 indicada por la figura de arriba, si se ejecuta la siguiente instrucción, ¿cuáles serían los nuevos valores de los registros del microprocesador? Rellenad la tabla siguiente para cada instrucción partiendo de la situación actual (no para las tres instrucciones seguidas).

		A	B	C	D	E	H	L
a)	**mov b,d**							
b)	**mov a,m**							
c)	**mov m,e**							

d) Una de estas tres instrucciones no modifica ningún registro, sino una posición de memoria. ¿Cuál es? ¿Cuál es la posición de memoria modificada? ¿Cuál es el cambio que experimenta el contenido de esa posición de memoria?

e) ¿Qué es el registro **F** (qué información contiene)? ¿Las operaciones indicadas arriba lo modifican? ¿Por qué?

f) ¿Qué es el registro **PC**? ¿Las operaciones indicadas arriba lo modifican? ¿Por qué y cómo?

2. Queremos llevar el dato de la posición de memoria 9003 a la posición 9008. Realiza esta operación con dos instrucciones.

3. Partiendo de la situación indicada en el ejercicio 1, explica qué ocurrirá al ejecutar las siguientes instrucciones (qué cambia y cómo). Suponer que se ejecutan todas las instrucciones seguidas.

   a) **sta** 9001

   b) **lda** 9001

   c) **shld** 9001

   d) **lhld** 9001

4. En el ejercicio 1, ¿qué pareja de registros se ha utilizado como apuntador de memoria? ¿Cómo se puede inicializar? Es decir, si queremos utilizar la posición 900A como apuntador de memoria, ¿qué operación debemos realizar?
   Nota: *Si tenemos un array de datos en memoria, podemos inicializar el apuntador con la dirección del primer elemento del array, y luego podemos recorrer todo el array incrementando en uno el contenido del apuntador.*

5. ¿Cuál es la diferencia entre las siguientes instrucciones? (Explica qué hace cada una)

   a) **lda addr** *(A) ← [addr]*

   b) **ldax rp** *(A) ← [(rp)]*
   c) **sta addr** *[addr] ← (A)*

   d) **stax rp** *[(rp)] ← (A)*

6. ¿Cuál es el resultado de ejecutar la instrucción **xchg**? ¿Para qué función resulta útil?

INSTRUCCIONES ARITMÉTICAS

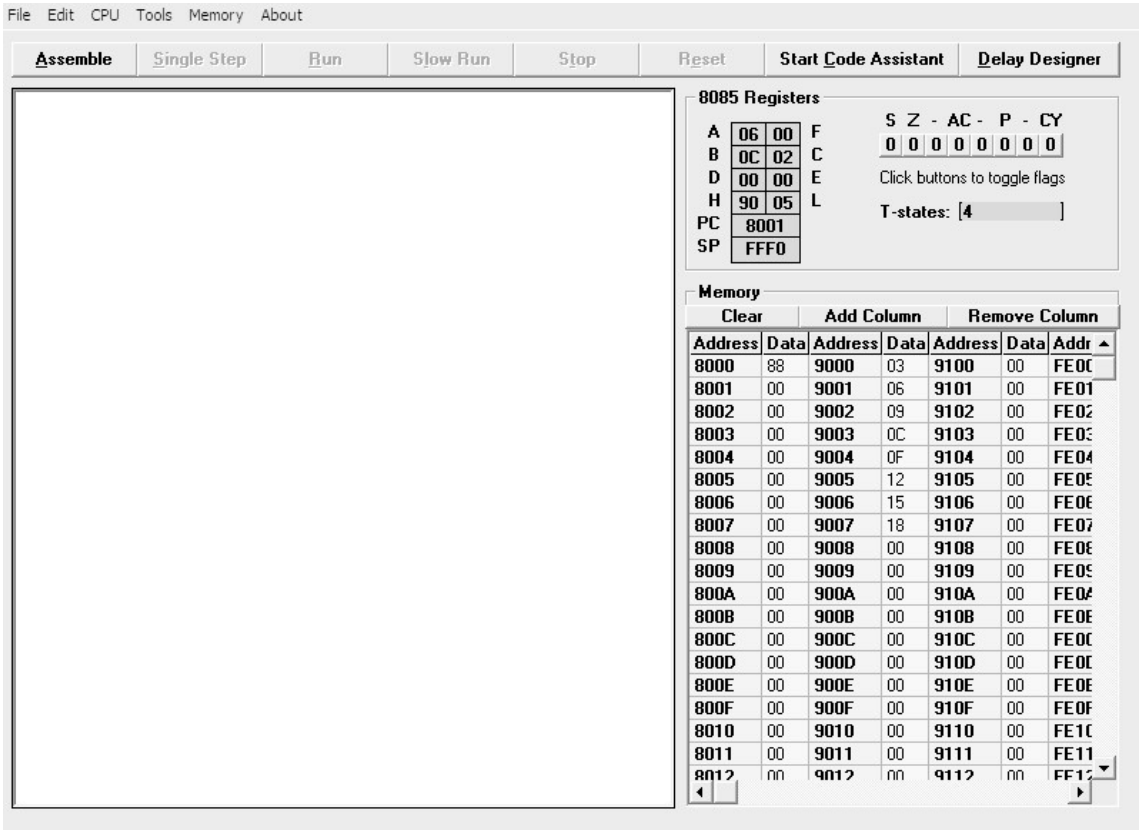
En la mayoría de las instrucciones aritméticas interviene el acumulador. En la instrucción se indica un operando mediante un dato, un registro o una posición de memoria de manera explícita, y para definir el otro operando, siempre se utiliza el acumulador. El resultado de la operación se escribe en el acumulador, de modo que se pierde el dato escrito en el acumulador antes de ejecutar la instrucción. Por tanto:

a) Si queremos volver a utilizar el dato contenido en el acumulador, debemos guardarlo antes de ejecutar la instrucción (en un registro o en una posición de memoria, mediante instrucciones de transferencia).
b) El resultado sólo se guarda en el acumulador, por lo que debemos copiarlo en otro sitio antes de perderlo por descuido en la siguiente instrucción.

A diferencia de las instrucciones de transferencia, las aritméticas activan casi todos los flag (indicadores de estado).

1. Explica lo que realizan las siguientes instrucciones y, suponiendo que la situación al comienzo es la que indica la imagen (la primera viene hecha a modo de ejemplo), ¿cuál sería el resultado que se escribiría en el acumulador?

NOTA: (no se ejecuta una instrucción tras otra)



a) ADD b

b) ADD c

c) ADD a

d) ADD m

e) ADI 08

f) ADC b

g) ADC c

h) ADC m

i) ACI 08

j) SUB b

k) SUB c

l) SUB a

m) SUB m

n) SUI 08

o) SBB b

p)SBB c

q) SBB m

r) SBI 08

2. A partir de la posición 9000h hay un array de 12 elementos. Supongamos que se establece un bucle que, con ayuda de un contador, va incrementando los elementos de array uno por uno. Escribe las instrucciones necesarias para realizar lo siguiente:

a) Inicializar la pareja de registros que usaremos como apuntador.

b) Incrementar dentro del bucle el i-ésimo elemento (el que toque) del array.

c) Actualizar el apuntador (para que apunte al siguiente elemento del array).

INSTRUCCIONES LÓGICAS

1) ¿Cuáles son las instrucciones lógicas que afectan a los flags? ¿A qué flags afectan?

2) a) Escribe la tabla de la verdad de las operaciones lógicas fundamentales AND, OR, XOR y NOT.

b) ¿Cómo se puede realizar una operación NOT mediante una puerta XOR?

3) En el 8085 el grupo de instrucciones lógicas lo forman AND, XOR, OR y CMP. CMP realiza la comparación entre un registro (CMP r), el contenido de una posición de memoria (CMP M) o un valor concreto (CPI byte) y el valor de acumulador. ¿Perdemos el valor contenido en el acumulador antes de la ejecución de la instrucción?¿Quién contiene el resultado?¿Cómo podemos utilizar el resultado de esta operación?

4) MÁSCARAS. Tenemos un número binario desconocido de 8 dígitos (XXXXXXXX2). Utilizando las operaciones lógicas:

a) ¿Cómo ponemos todos los bits a '1' (11111111_2)?

b) ¿Cómo ponemos todos los bits a '0' (00000000_2)?

c) ¿Cómo ponemos los cuatros bits más significativos a '1', sin cambiar el valor de los otros cuatros bits?

d) ¿Cómo ponemos los cuatro bits más significativos a '0', sin cambiar el valor de los otros cuatros bits?

e) ¿Cómo ponemos todos los bits a '0' salvo el MSB (éste se mantiene como estaba)?¿Cuáles son los resultados posibles de esta operación?

5) Supongamos que el número binario desconocido del ejercicio anterior es 100111012. Escribe en hexadecimal cuáles deben ser las máscaras y los resultados para los apartados de a) a e) del ejercicio anterior.

6) Con el dato guardado en la posición de memoria 9002, realiza las siguientes operaciones:

a) Anular todos sus bits y guardar el resultado en la posición 9003.

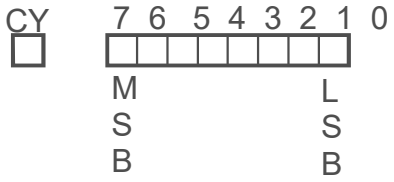
b) Sin cambiar el LSB, pon todos sus otros bits a '1' y guarda el resultado en la posición 9004.

c) Sin cambiar el MSB, pon todos sus otros bits a '0' y guarda el resultado en la posición 9005.

7) Se quiere saber si el dato de la posición de memoria 9100 es mayor o menor que el de la posición de memoria 9101. ¿Qué podemos hacer (empleando operaciones lógicas) para saberlo? Repetir (con el dato de la posición 9100) para el dato que contiene el registro B y para el número 12_{10}.

INSTRUCCIONES DE ROTACIÓN

1) Representa el efecto que tienen las instrucciones RLC, RRC, RAL y RAR en los bits del registro.



2) El dato que rota, ¿en qué registro está?

3) El contenido del registro es 40h. ¿Cuál es la diferencia entre realizar dos instrucciones RLC seguidas y ejecutar dos instrucciones RAL seguidas? Repetir si el contenido del registro es 01h y las instrucciones que se ejecutan seguidas (una vez) son RRC y RAR.

4) Si el contenido del registro es 10001011_2, ¿cuál es su valor decimal?
¿Cuál será el resultado de ejecutar RLC y RRC (en base binaria y en base decimal)?
¿Y si el contenido es 00010000_2 (valor decimal y resultados de las rotaciones)?
Si en lugar de rotar realizamos desplazamiento y en las posiciones del bit entrante escribimos '0', ¿qué conseguimos si desplazamos un número binario una posición a la izquierda? ¿Y si los desplazamos a la derecha?

5) ¿Cuál es el complemento a uno de un número (definición y método práctico)? ¿Y el complemento a dos?

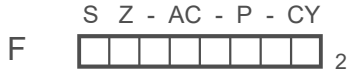
6) Diferencias entre las instrucciones CMA y CMC (explica qué hace cada una).

INSTRUCCIONES DE BIFURCACIÓN

1-Rellena la siguiente tabla con el significado de los valores de los bits de estado (flags).

Bit de estado	Valor	Significado: el resultado de la operación anterior es…
S	0	
S	1	
Z	0	
Z	1	
P	0	
P	1	
CY	0	
CY	1	

2- ¿Cómo ha sido el resultado de la operación anterior si el valor del registro F es el siguiente?



a) F=04h

b) F=11h

c) F=C4h

3- Si el resultado de la anterior operación es negativo, salta a la etiqueta NEGATIVO

4- Si el resultado es negativo, salta a NEG; si no salta a NO_NEG

EJERCICIOS COMPLETOS

1- ¿Qué hace el siguiente programa?

```
mvi a,00
bucle:
lda 9000
cpi 00
jz fin
jm negativo
jmp bucle
negativo:
;aquí habrá unas operaciones que omitimos
fin:
```

2- ¿Qué hace el siguiente programa?

```
mvi a,00
bucle:
lda 9000
mov b,a
lda 9001
sub b
jz fin
jmp diferentes
diferentes:
;aquí habrá unas operaciones que omitimos
fin:
```

3- En la posición 9000h está el primer elemento de un array. Este array es de 10 elementos y todos son positivos. Escribe un programa que recorra todos los elementos del array decrementando en 1 su valor. Cuando llegue al final del array empezará de nuevo, decrementando en otra unidad todos los elementos del array. El programa se repetirá hasta que algún elemento llegue a cero.

4- Explica qué hace el siguiente programa, cada instrucción y el conjunto.

```
lhld 9100
mvi b, 00
mvi c, 10
mvi d, 08
bucle:
mov a,d
cmp m
jm siguiente
inr b
siguiente:
inx hl
dcr c
jz fin
jmp bucle
fin:
```

Grado en Ingeniería Informática de Gestión y Sistemas de Información
Estructura de Computadores
Pre-laboratorio (ejercicios para casa): SOLUCIONES

REPASO

En el 8086 se utiliza la **base hexadecimal**.

a) [Sol]: puede tomar 16 valores diferentes: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

b) [Sol]: 4 cifras binarias.

c) [Sol]: Los registros tienen 8 bits, por tanto 2 cifras hexadecimales; El valor máximo que pueden contener será: $(FF)_{16}$, $(1111\ 1111)_2$ y $(255)_{10}$

d) [Sol]: Se multiplexan en el tiempo el bus de datos (8 bits) y el bus de direcciones (8 bits) para poder direccionar 64K posiciones de memoria. Son necesarios **2 registros** para usarlos como apuntador de memoria.

INSTRUCCIONES DE TRANSFERENCIA

1.

		A	B	C	D	E	H	L
a)	**mov b,d**	05	45	0A	45	54	90	05
b)	**mov a,m**	05	90	0A	45	54	90	05
c)	**mov m,e**	05	90	0A	45	54	90	05

d) [Sol]: La instrucción c) sólo modifica la posición de memoria apuntada por el HL, en este caso la 9005. En esa posición de memoria se escribirá el contenido del registro E, es decir, 54.

e) ¿[Sol]: El registro F contiene la información de los flags de estado (S, Z, AC, P, CY). Las instrucciones indicadas en este ejercicio son todas de transferencia, por lo que no realizan ninguna operación que modifique los flags de estado.

f) [Sol]: El registro PC contiene la dirección de la siguiente instrucción a ejecutar. Por tanto, ninguna de las instrucciones de este ejercicio lo modifica directamente. Sin embargo, la ejecución de cualquier instrucción provoca el cambio de este registro automáticamente, de modo que apunte a la dirección de la siguiente instrucción después de haberse ejecutado la actual.

2.
[Sol]: **lda 9003** (A)←[9003]
sta 9008 [9008]←(A)

3.
a) **sta** 9001
[Sol]: **[9001]←(A);** El contenido del registro A, 05h, se escribe en la dirección de memoria 9001h (sólo cambia la memoria).

b) **lda** 9001
[Sol]: **(A)←[9001];** El contenido de la dirección de memoria 9001h, 01h, se escribe en el registro A (sólo cambia el registro A).

c) **shld** 9001
[Sol]: **[9001]←(L) y [9002]←(H);** El contenido del registro L , 05h, se escribe en la dirección de memoria 9001h y el contenido del registro H , 90h, se escribe en la dirección de memoria 9002h (sólo cambia la memoria).

d) **lhld** 9001
[Sol]: **(L)←[9001] eta (H)←[9002],** El contenido de la dirección de memoria 9001h, 01h se escribe en el registro L y el contenido de la dirección de memoria 9002h, 02h, se escribe en el registro H (sólo cambian los registros H y L).

4.
[Sol]: Se ha utilizado la pareja de registros HL. Para inicializarla:
lxi h, 900A (HL)←900Ah

5.
a) **lda addr** *(A) ← [addr]*
[Sol]: Escribe en el registro A, el contenido de la posición de memoria de dirección addr.

b) **ldax rp** *(A) ← [(rp)]*
[Sol]: Escribe en el registro A, el contenido de la posición de memoria cuya dirección está en la pareja de registros rp;

c) **sta addr** *[addr] ← (A)*
[Sol]: Escribe en memoria en la posición de dirección addr, el contenido del registro A.

d) **stax rp** *[(rp)] ← (A)*
[Sol]: Escribe en memoria, en la posición de memoria cuya dirección está en el registro doble rp, el contenido del registro A.

Por tanto, en los casos a y c se indica directamente la dirección de memoria, mientras que en los casos b y d se emplea un registro doble como puntero a una dirección de memoria.

6.
[Sol]: La instrucción xchg intercambia el contenido de las parejas de registros HL y DE. Esta operación resulta útil cuando utilizamos dos apuntadores de memoria diferentes mediante las dos parejas de registros. Como la pareja HL es la única que se puede utilizar en las instrucciones mov, podemos intercambiar los apuntadores para los accesos a memoria y luego reintegrarlos a los valores anteriores, sin perder ninguno de los dos.

(H)←→(D) y **(L)←→(E)**

INSTRUCCIONES ARITMÉTICAS

1.
a) ADD b
[Sol]: (A) ← (A)+(B); 12h ←06h+0Ch; En el acumulador había 06h, en el registro B había 0Ch (12_{10}), por tanto, el resultado que se queda en el acumulador es: ($6_{10}+12_{10}=18_{10}$) =12h.

b) ADD c
[Sol]: (A) ← (A)+(C); 08h ← 06h+02h; En el acumulador había 06h, en el registro C había 02h (2_{10}), por tanto, el resultado que se queda en el acumulador es: ($6_{10}+2_{10}=8_{10}$) =08h.

c) ADD a
[Sol]: (A) ← (A)+(A); 0Ch ←06h+06h; En el acumulador había 06h, por tanto, el resultado que se queda en el acumulador es:($6_{10}+6_{10}=12_{10}$) =0Ch.

d) ADD m
[Sol]: (A) ← (A)+[(HL)]; 18h ←06h+12h; En el acumulador había 06h , en la posición de memoria apuntada por el registro HL (9005h), había 12h (18_{10}), por tanto, el resultado que se queda en el acumulador es:($6_{10}+18_{10}=24_{10}$) =18h.

e) ADI 08
[Sol]: (A) ← (A)+08h; 0Eh ←06h+08h; En el acumulador había 06h, por tanto, el resultado que se queda en el acumulador es:($6_{10}+8_{10}=14_{10}$) =0Eh.

f) ADC b
[Sol]: (A) ← (A)+(B)+(CY); 12h ←06h+0Ch+0b; En el acumulador había 06h , en el registro B había 0Ch (12_{10}) y el el bit de carry había 0b, por tanto, el resultado que se queda en el acumulador es:($6_{10}+12_{10}+0_{10}=18_{10}$) =12h.

g) ADC c
[Sol]: (A) ← (A)+(C)+(CY); 08h ←06h+02h+0b; En el acumulador había 06h , en el registro C había 02h y el el bit de carry había 0b, por tanto, el resultado que se queda en el acumulador es:($6_{10}+2_{10}+0_{10}=8_{10}$) =08h.

h) ADC m
[Sol]: (A) ← (A)+[(HL)]+(CY); 18h ←06h+12h+0b; En el acumulador había 06h, en la posición de memoria apuntada por el registro HL (9005h), había 12h (18_{10}) y el el bit de carry había 0b, por tanto, el resultado que se queda en el acumulador es:($6_{10}+18_{10}+0_{10}=24_{10}$) =18h.

i) ACI 08
[Sol]: (A) ← (A)+08h+(CY); 0Eh ←06h+08h+0b; En el acumulador había 06h y el el bit de carry había 0b, por tanto, el resultado que se queda en el acumulador es: ($6_{10}+8_{10}+0_{10}=14_{10}$) =0Eh.

j) SUB b
[Sol]: (A) ← (A)-(B); FAh ←06h-0Ch; En el acumulador había 06h, en el registro B había 0Ch (12_{10}), por tanto, el resultado que se queda en el acumulador es:($6_{10}-12_{10}=-6_{10}$) =FAh.

k) SUB c
[Sol]: (A) ← (A)-(C); 04h ←06h-02h; En el acumulador había 06h, en el registro C había 02h, por tanto, el resultado que se queda en el acumulador es:($6_{10}-2_{10}=4_{10}$) =04h.

l) SUB a
[Sol]: (A) ← (A)-(A); 00h ←06h-06h; En el acumulador había 06h, por tanto, el resultado que se queda en el acumulador es:($6_{10}-6_{10}=0_{10}$) =00h.

m) SUB m
[Sol]: (A) ← (A)-[(HL)]; F4h ←06h-12h; En el acumulador había 06h, en la posición de memoria apuntada por el registro HL (9005h), había 12h (18_{10}), por tanto, el resultado que se queda en el acumulador es:($6_{10}-18_{10}=-12_{10}$) =F4h.

n) SUI 08
[Sol]: (A) ← (A)-08h; FEh ←06h-08h; En el acumulador había 06h, por tanto, el resultado que se queda en el acumulador es:($6_{10}-8_{10}=-2_{10}$) =FEh.

o) SBB b
[Sol]: (A) ← (A)-(B)-(CY); FAh ←06h-0Ch-0b; En el acumulador había 06h, en el registro B había 0Ch (12_{10}) y el el bit de carry había 0b, por tanto, el resultado que se queda en el acumulador es:($6_{10}-12_{10}-0_{10}=-6_{10}$) =FAh.

p)SBB c
[Sol]: (A) ← (A)-(C)-(CY); 04h ← 06h-02h-0b; En el acumulador había 06h, en el registro C había 02h y el el bit de carry había 0b, por tanto, el resultado que se queda en el acumulador es:($6_{10}-2_{10}-0_{10}=4_{10}$) =04h.

q) SBB m

[Sol]: (A) ← (A)-[(HL)]-(CY); F4h ← 06h-12h-0b; En el acumulador había 06h , en la posición de memoria apuntada por el registro HL (9005h), había 12h (18_{10}) y en el bit de carry había 0b, por tanto, el resultado que se queda en el acumulador es:(6_{10}-18_{10}-0_{10}=-12_{10}) =F4h.

r) SBI 08
[Sol]: (A) ← (A)-08h-(CY); FEh ← 06h-08h-0b; En el acumulador había 06h y en el bit de carry había 0b, por tanto, el resultado que se queda en el acumulador es: (6_{10}-8_{10}-0_{10}=-2_{10}) =FEh.

2.
a) [Sol]: Escogemos, por ejemplo, la pareja de registros HL, por lo que cargaremos en ella el valor de la primera posición del array:

lxi h, 9000 (HL) ← 9000h

b) [Sol]: Si el apuntador está actualizado, el HL contiene la dirección del pésimo elemento, por tanto, se puede incrementar así:

inr m [(HL)] ← [(HL)]+1

c) [Sol]: Si el actual valor del apuntador es el correspondiente a la posición de memoria del elemento actual del array, como todos los elementos del array están en posiciones consecutivas de memoria, basta con aumentar en uno el valor del apuntador HL para que apunte al siguiente elemento del array:

inx h (HL) ← (HL)+1

INSTRUCCIONES LÓGICAS

1) [Sol]: Todas las instrucciones lógicas afectan a **todos los flags.**

2) a)[Sol]:

AND		
X	Y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR		
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

NOT	
X	Y
0	1
1	0

2) b) [Sol]: En la tabla de la verdad de XOR se puede ver que, si la x es 1, la F es siempre la negada de la y, por tanto basta con asegurar que **uno de los operandos de la puerta XOR sea siempre 1.**

3) [Sol]: El resultado de la instrucción CMP no se guarda, por lo que **el valor contenido en el acumulador no cambia** después de la ejecución de la instrucción. El único efecto de la ejecución de la instrucción es la modificación de todos **los flags de estado**, por tanto en ellos aparece el valor del resultado. Los valores de los flags de estado definen el resultado de las **instrucciones de salto condicional**.

4)

a) [Sol]: Como A+1=1, nos basta con hacer **ORI FFh** (FF_{16})=(11111111_2).
b) [Sol]: Como A·0=0, hacemos **ANI 00h** (00_{16})=(00000000_2).
c) [Sol]: Como A+1=1 y A+0=A, hacemos **ORI F0h** ($F0_{16}$)=(11110000_2).
d) [Sol]: Como A·0=0 y A·1=A, hacemos **ANI 0Fh** ($0F_{16}$)=(00001111_2).
e) [Sol]: Como A·0=0 y A·1=A, hacemos **ANI 80h** (80_{16})=(10000000_2).
Si MSB=0, el resultado es **00h**, pero si MSB=1, el resultado es **80h**.

5) [Sol]:

a) Máscara FFh, resultado FFh.
b) Máscara 00h, resultado 00h.
c) Máscara F0h, resultado FDh.
d) Máscara 0Fh, resultado 0Dh.
e) Máscara 80h, resultado 80h

6) [Sol]:

a)
```
LDA 9002h
ANI 00h
STA 9003h
```

b)
```
LDA 9002h
ORI FEh
STA 9004h
```

c)
```
LDA 9002h
ANI 80h
STA 9005h
```

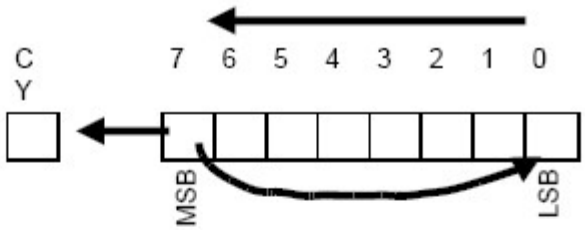
7) [Sol]:

```
a) lda 9100          (A) ← [9100]
   lxi h, 9101       (HL) ← 9101
   cmp m             (A)-[(HL)]

b) lda 9100          (A) ← [9100]
   cmp b             (A)-(B)

c) lda 9100          (A) ← [9100]
   cpi 0C            (A) – 0Ch (0Ch=12₁₀)
```

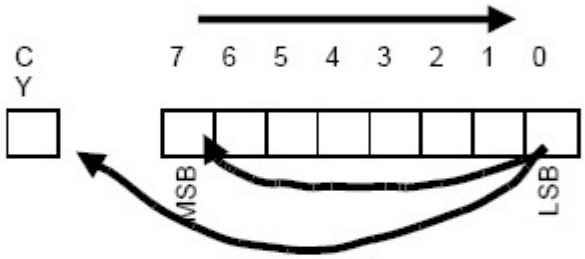
INSTRUCCIONES DE ROTACIÓN

1) [Sol]:

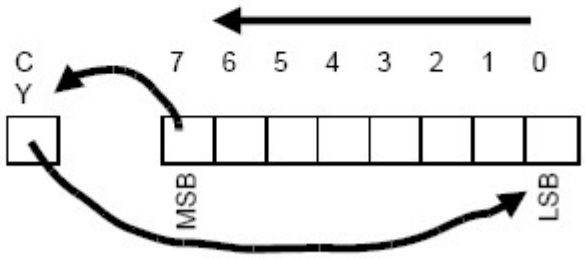
RLC



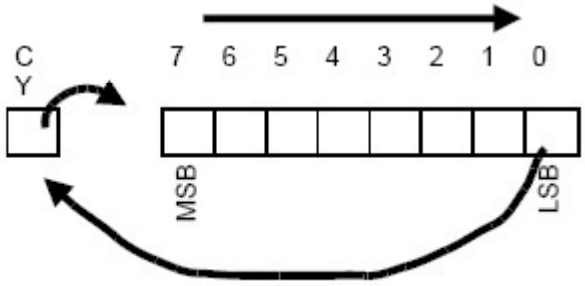
RRC



RAL



RAR



2) [Sol]: En el acumulador, en el registro A.

3) El contenido del registro es 40h. ¿Cuál es la diferencia entre realizar dos instrucciones RLC seguidas y ejecutar dos instrucciones RAL seguidas? Repetir si el contenido del registro es 01h y las instrucciones que se ejecutan seguidas (una vez) son RRC y RAR.
[Sol]:

$A=01000000_2=40_{16}=64_{10}$

RLC $A=10000000_2=80_{16}=128_{10}$ CY=0

RLC $A=00000001_2=01_{16}=01_{10}$ CY=1

$A=01000000_2=40_{16}=64_{10}$ Suponiendo que CY =0 antes de ejecutarlas:

RAL $A=10000000_2=80_{16}=128_{10}$ CY=0

RAL $A=00000000_2=00_{16}=0_{10}$ CY=1

$A=00000001_2=01_{16}=1_{10}$

RRC A=10000000_2=80_{16}=128_{10} CY=1
RAR A=11000000_2=$C0_{16}$=192_{10} CY=0

4) Si el contenido del registro es 10001011_2, ¿cuál es su valor decimal?
¿Cuál será el resultado de ejecutar RLC y RRC (en base binaria y en base decimal)?
¿Y si el contenido es 00010000_2 (valor decimal y resultados de las rotaciones)?
Si en lugar de rotar realizamos desplazamiento y en las posiciones del bit entrante escribimos '0', ¿qué conseguimos si desplazamos un número binario una posición a la izquierda? ¿Y si los desplazamos a la derecha?

[Sol]:

A=100010112=8B16=13910
RLC A=000101112=1716=**2310 CY=1**
RRC A=110001012=C516=**19710CY=1**

A=000100002=1016=1610
RLC A=001000002=2016=**3210 CY=0**
RRC A=000010002=0816=**810 CY=0**

Al desplazar una posición a la izquierda, todos los bits pasan a ocupar una posición de mayor peso, el resultado es que el número se multiplica por dos, pero como el MSB desaparece, hay que restar ese valor: **shl (N)= $2 \cdot N - b_{n-1} \cdot 2_n$**

Al desplazar una posición a la derecha, todos los bits pasan a ocupar una posición de menor peso, el resultado es que el número se divide entre dos, pero como el lsb desaparece, hay que restar ese número: **shr (N)= $N/2 - b_0 \cdot 2_{-1}$**

5) ¿Cuál es el complemento a uno de un número (definición y método práctico)? ¿Y el complemento a dos?
[Sol]:
Complemento a uno de A:
$2^n-1-|A|$; n: número de bits de A.
Método práctico: **Invertir todos los bits de A.**

Complemento a dos de A:
$2^n-|A|$; n: número de bits de A.
Método práctico: **Sumar uno al complemento a uno de A.**

6) Diferencias entre las instrucciones CMA y CMC (explica qué hace cada una).

[Sol]:
La instrucción **CMA** realiza el **NOT** de todos los bits contenidos en el **registro A.**
La instrucción **CMC** sólo realiza el **NOT** del contenido del **flag de carry CY.**

INSTRUCCIONES DE BIFURCACIÓN

1-Rellena la siguiente tabla con el significado de los valores de los bits de estado (flags).

Bit de estado	Valor	Significado: el resultado de la operación anterior es…
S	0	Positivo
S	1	Negativo
Z	0	Distinto de 0
Z	1	Cero
P	0	Número impar de unos
P	1	Número par de unos
CY	0	Sin acarreo de la última cifra
CY	1	Con acarreo de la última cifra

2-

a) F=04h
[Sol]: F=04h=00000100_2 ; S=0; Z=0; AC=0; P=1; CY=0;
El resultado tiene un número par de 1s, es positivo y distinto de cero y no hay acarreo.

b) F=11h
[Sol]: F=11h=00010001_2 ; S=0; Z=0; AC=1; P=0; CY=1;
El resultado tiene un número impar de 1s, es positivo y distinto de cero y hay acarreo de la última cifra y de la cuarta.

c) F=C4h
[Sol]: F=C4h=11000100_2 ; S=1; Z=1; AC=0; P=1; CY=0;
No es posible que S y Z estén a uno al mismo tiempo.

3- [Sol]:
```
jm NEGATIVO
 (…)
NEGATIVO:
```

4- [Sol]:
```
jm NEG
jmp NO_NEG
 (…)
NEG:
(…)
NO_NEG:
(…)
```

EJERCICIOS COMPLETOS

1-

```
mvi a,00        Se inicializa el acumulador a cero,
bucle:
lda 9000        se lee el contenido de 9000h [a←(9000)]
cpi 00          y se compara con cero
jz fin          si son iguales, fin del programa
jm negativo     si es menor que cero, salta a negativo
jmp bucle       si es positivo, vuelve a ejecutar el bucle
negativo:
;aquí habrá unas operaciones que omitimos
fin:
```

El programa comprueba el dato en la posición 9000h mientras sea positivo; si es negativo ejecutará una parte de código (omitido), y si es cero, termina el programa.

2-

```
mvi a,00            El acumulador se inicializa a cero,
bucle:
lda 9000            se lee el dato de 9000 al acumulador,
mov b,a             lo guarda en b,
lda 9001            se lee el dato de 9001 al acumlador,
sub b               se comparan (a←a-b) y
jz fin              si son iguales fin del programa,
jmp diferentes      en caso contrario salta a diferentes
diferentes:
;aquí habrá unas operaciones que omitimos
fin:
```

El programa compara los datos en las posiciones 9000h y 9001h y ejecuta un código (aquí omitido) solo si son diferentes.

3- [Sol]:

```
inicio:
lxi hl,9000
mvi b,0a
bucle:
dcr m
jz fin
inx hl
dcr b
jnz bucle
jmp inicio
fin:
```

4- [Sol]:

```
lhld 9100 ; contenido de dirección 9100 al registro L; y la de 9101 al registro H
mvi b, 00 ; inicializo registro b a 0
mvi c, 10 ; inicializo registro c a 10
```

```
mvi d, 08 ; incializo registro d a 08
bucle:
mov a,d ; (a)← (d)=08h
cmp m ; (a)-[(HL)] comparación: ; el contenido del acumulador (8) y el contenido del primer elemento del array que empieza en la posición indicada en 9100 y 9101
jm siguiente; si el resultado es negativo (8 > X); salta a “siguiente” sin incrementar b
inr b ;si no (8 < X) incrementa b
siguiente:
inx hl ; incrementa el apuntador
dcr c ; decrementa el registro c
jz fin ; si el contenido de c es cero salta a “fin”
jmp bucle ; de lo contrario repite el bucle
fin:
```

La dirección de memoria formada por los dos bytes contenidos en las posiciones 9100h y 9101h es la del primer elemento de un array. Este array tiene 16 elementos y por eso introducimos 10h en C, para usarlo como contador al recorrer el array. El registro B también será un contador, en este caso muestra el número de elementos del array que son menores que 08.

Diagrama de bloques del 8085

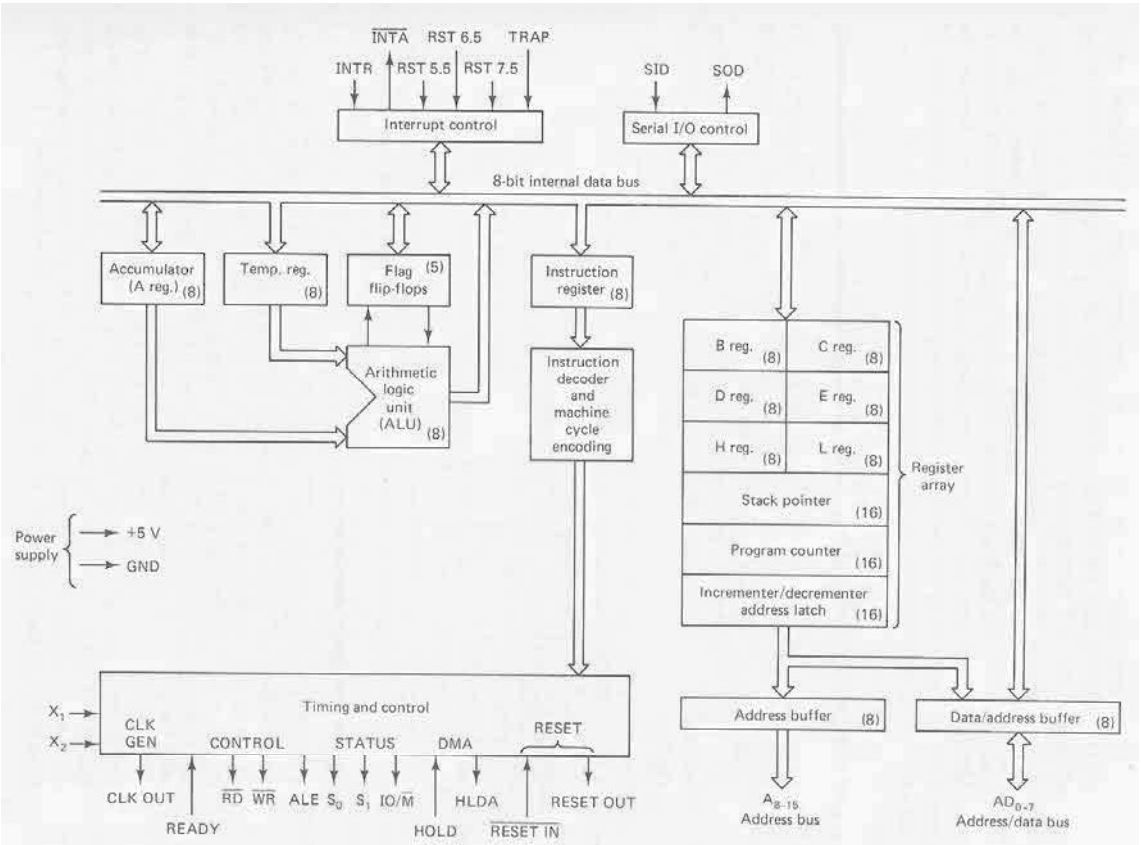
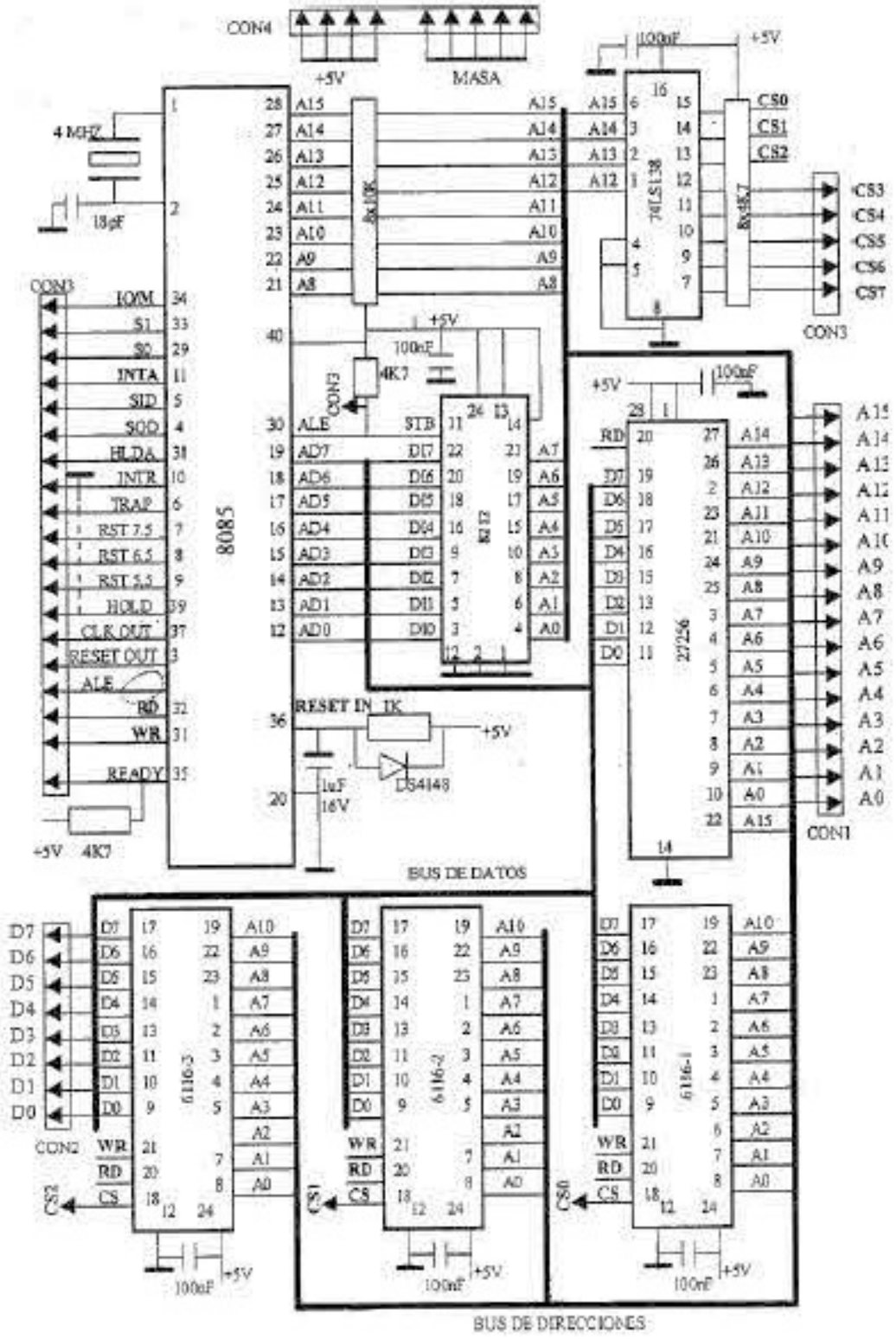


Figure 3–1 The functional block diagram of the Intel 8085A microprocessor. (Diagram courtesy of Intel Corporation.)

Unidad de control y registros del 8085



Juego de instrucciones del 8085

NEMONICO	EXP. GRAF.	FLAGS
INSTRUCCIONES DE TRANSFERENCIA		
MOV r1,r2	(r1)←(r2)	NINGUNO
MOV r,M	(r)←[(HL)]	NINGUNO
MOV M,r	[(HL)] ←(r)	NINGUNO
MVI r,byte	(r)←byte	NINGUNO
MVI M,byte	[(HL)]← byte	NINGUNO
LXI rp,doble	(rpl)←1º byte (rph)←2º byte	NINGUNO
LDA addr	(A)←[addr]	NINGUNO
STA addr	[addr]←(A)	NINGUNO
LHLD addr	(L)←[addr] (H)←[addr+1]	NINGUNO
SHLD addr	[addr]←(L) [addr+1]←(H)	NINGUNO
LDAX rp	(A)←[(rp)]	NINGUNO
STAX rp	[(rp)]←(A)	NINGUNO
XCHG	(H)↔(D) (L)↔(E)	NINGUNO
INSTRUCCIONES ARITMÉTICAS		
ADD r	(A)←(A)+(r)	TODOS
ADD M	(A)←(A)+[(HL)]	TODOS
ADI byte	(A)←(A)+byte	TODOS
ADC r	(A)←(A)+(r)+CY	TODOS
ADC M	(A)←(A)+[(HL)]+CY	TODOS
ACI byte	(A)←(A)+byte+CY	TODOS
SUB r	(A)←(A)-(r)	TODOS
SUB M	(A)←(A)-[(HL)]	TODOS
SUI byte	(A)←(A)-byte	TODOS
SBB r	(A)←(A)-(r)-CY	TODOS
SBB M	(A)←(A)-[(HL)]-CY	TODOS
SBI byte	(A)←(A)-byte-CY	TODOS
INR r	(r)←(r)+1	Z, S, P, AC
INR M	[(HL)]←[(HL)]+1	Z, S, P, AC
DCR r	(r)←(r)-1	Z, S, P, AC
DCR M	[(HL)]←[(HL)]-1	Z, S, P, AC
INX rp	(rp)←(rp)+1	NINGUNO
DCX rp	(rp)←(rp)-1	NINGUNO
DAD rp	(HL)←(HL)+(rp)	CY
DAA	Ajuste BCD de (A)	NINGUNO
INSTRUCCIONES LÓGICAS.		
ANA r	(A)←(A) and (r) (CY)←0, (AC)←1	TODOS
ANA M	(A)←(A) and [(HL)] (CY)←0, (AC)←1	TODOS
ANI byte	(A)←(A) and byte (CY)←0, (AC)←1	TODOS
XRA r	(A)←(A) xor (r)	TODOS
XRA M	(A)←(A) xor [(HL)]	TODOS
XRI byte	(A)←(A) xor byte	TODOS
ORA r	(A)←(A) or (r)	TODOS
ORA M	(A)←(A) or [(HL)]	TODOS
ORI byte	(A)←(A) or byte	TODOS
CMP r	(A)-(r)	TODOS
CMP M	(A)-[(HL)]	TODOS
CPI byte	(A)-byte	TODOS

NEMONICO	EXP. GRAF.	FLAGS
INSTRUCCIONES DE ROTACIÓN Y FLAGS		
RLC	Rotacion izqda	CY
RRC	Rotación dcha	CY
RAL	Rot. izqda. con CY	CY
RAR	Rot. dcha. con CY	CY
CMA	Comp. A1 de (A)	NINGUNO
CMC	Invierte (CY)	CY
STC	(CY)←1	CY
INSTRUCCIONES DE BIFURCACIÓN ccc=NZ salto si no cero (Z=0), ccc=Z salto si cero (Z=1), ccc=NC salto si no acarreo (CY=0), ccc=C salto si acarreo (CY=1) , ccc=PO salto si paridad impar (P=0), ccc=PE salto si paridad par (P=1), ccc=P salto si positivo (S=0), ccc=M salto si negativo (S=1)		
JMP addr	(PC)←addr	NINGUNO
Jccc addr	Si ccc=1, (PC)←addr; Si ccc=0, (PC)←(PC)+3	NINGUNO
CALL addr	Guarda PC en la pila (PC)←addr	NINGUNO
Cccc addr	Si ccc=1, guarda PC en la pila, (PC)←addr; Si ccc=0, (PC)←(PC)+3	NINGUNO
RET	Recupera PC de la pila	NINGUNO
Rccc	Si ccc=1, recupera PC de la pila; Si ccc=0 (PC)←(PC)+1	NINGUNO
RSTn	(PC)←n x 8	NINGUNO
PCHL	(PC)←(HL)	NINGUNO
INSTRUCCIONES DE MANEJO DE LA PILA.		
PUSH rp	[(SP)-1]←(rpl) [(SP)-2]←(rph) (SP)←(SP)-2	NINGUNO
PUSH PSW	[(SP)-1]←(A) [(SP)-2]←(RE) (SP)←(SP)-2	NINGUNO
POP rp	(rph) ←[(SP)] (rpl)← [(SP)+1] (SP)←(SP)+2	NINGUNO
POP PSW	(RE) ←[(SP)] (A)← [(SP)+1] (SP)←(SP)+2	NINGUNO
XTLH	(L) ↔[(SP)] (H)↔ [(SP)+1]	NINGUNO
SPLH	(HL) ↔(SP)	NINGUNO
INSTRUCCIONES DE ENTRADA Y SALIDA.		
IN puerta	(A)←[puerta]	NINGUNO
OUT puerta	[puerta]←(A)	NINGUNO
INSTRUCC. DE CONTROL DE INTERRUPCIONES		
EI	Habilita interrupciones	NINGUNO
DI	Inhabilita interrupciones	NINGUNO
HLT	Para el microprocesador	NINGUNO
NOP	No hace nada	NINGUNO
RIM	Lee linea serie y estado interrupciones.	NINGUNO
SIM	Escribe en linea serie y Programa interrupciones.	NINGUNO

RE=Status Register
PSW=Processor Status Word

Grado en Ingeniería Informática de Gestión y Sistemas de Información
Estructura de Computadores
L2

Implementa la UAL que diseñaste en las prácticas de aula y comprueba su funcionamiento.

Diseña una UAL que teniendo como entrada los números A y B de cuatro bits (A[A3, A2, A1, A0] y B[B3, B2, B1, B0]), realice las siguientes operaciones:

Aritméticas: A; incremento de A (A+1); suma (A+B); resta (A-B)
Lógicas: B; AND (A·B); OR (A+B); XOR (A⊕B)

Dibujar la tabla y el circuito para las operaciones aritméticas, y la tabla y el circuito de una etapa para las operaciones lógicas. ¿Cuántos bits de selección has necesitado? ¿Puedes reducir ese número?

Circuitos integrados:

7400	quad 2-input NAND gates
7402	quad 2-input NOR gates
7404	hex inverting gates
7408	quad 2-input AND gates
7410	triple 3-input NAND gates
7411	triple 3-input AND gates
7427	triple 3-input NOR gates
7432	quad 2-input OR gates
7486	quad 2-input exclusive-OR gates
7483	4-bit binary adder with fast carry
7485	4-bit magnitude comparator
74138	3-line to 8-line decoder/demux
74139	2 fully independent 2-to-4-line decoders/demultiplexers
74153	dual 1-of-4 line data selectors/mux
74154	4-line to 16-line decoder/demux

NOTA: Una copia de las hojas de características (datasheets) de los circuitos integrados están disponibles para consulta en el laboratorio.

1. Coge el dato de la posición de memoria 9101h y guárdalo en el acumulador.

2. Guarda el dato contenido en el acumulador en la posición de memoria 9000h

3. Carga el dato de la posición de memoria 9000h en el registro L y el dato de la posición de memoria 9001h en el registro H

4. Guarda el contenido del registro L en la posición de memoria 9000h y el contenido del registro H en la posición de memoria 9001h

5. Llevar el dato de la posición de memoria 9001h a la posición de memoria 9101h

6. Carga en el acumulador el contenido de la posición de memoria cuya dirección está en la pareja de registros B,C (empleando los registros BC como punteros)

7. Guarda en la posición de memoria, cuya dirección está en la pareja de registros B,C, el contenido del registro A

8. Intercambia el contenido de las parejas de registros HL y DE

9. Guarda el contenido del registro B en el registro E

10. Guarda el contenido de la posición de memoria apuntada por los registros HL, en el registro A

11. Guarda en la posición de memoria apuntada por los registros HL, el contenido del registro C

12. Supongamos que se ejecutan las siguientes instrucciones: mov E,B; mov A,m y mov m,C ¿Modifican estas instrucciones al registro F?, ¿Y al registro PC?,¿Por qué?

13. Guarda el dato 05h en el acumulador

14. Inicializar los registros HL como punteros a la posición de memoria 900Ah

15. Guardar el dato 05h en la posición de memoria 9001h sin emplear punteros

16. Guardar el dato 05h en la posición de memoria 9001h empleando punteros

17. Rellena la tabla con los valores que se cargan en memoria a partir de la posición 8000h al ensamblar el siguiente trozo de código. Ejecútalo paso a paso y explica qué ocurre con el registro PC, relacionando las instrucciones con los valores en memoria.

```
mvi B, 06
mov B, A
lda 9000
```

Address	Data
8000	
8001	

Grado en Ingeniería Informática de Gestión y Sistemas de Información
Estructura de Computadores
L4

Instrucciones aritméticas

1. Suma el contenido del Registro B al Acumulador y a ese valor Réstale el valor 7_{10} (07h)

2. Suma el contenido de la posición de memoria apuntada por los registros HL al acumulador y a ese valor Réstale el contenido del Registro B

3. Suma el dato 12_{10} (0Ch) al contenido del acumulador, y a ese valor Réstale el contenido del registro C

4. Resta el contenido de la posición de memoria apuntada por los registros HL al acumulador y a ese valor Súmale el dato 0Ah

5. Decrementa en 1 el contenido del Registro B e incrementa en 1 el contenido del Registro C

6. Suma 1 al contenido de la posición de memoria apuntada por los registros HL

7. Resta 1 al contenido de la posición de memoria apuntada por los registros HL

8. Suma el contenido del Registro B más la llevada de la operación anterior al acumulador y luego incrementa en 1 el contenido del registro B

9. Suma 9_{10} (09h) más la llevada de la operación anterior al Acumulador y luego réstale 1 el contenido del registro C

10. Resta el contenido del Registro B más la llevada de la operación anterior al acumulador y luego incrementa en 1 el contenido del registro C

11. Resta 2_{10} (02h) más la llevada de la operación anterior al Acumulador y luego réstale 1 el contenido del registro C

12. Suma el contenido de la posición de memoria apuntada por los registros HL más la llevada de la operación anterior al acumulador.

13. Resta el contenido de la posición de memoria apuntada por los registros HL más la llevada de la operación anterior al acumulador.

14. Incrementa en una unidad el contenido del registro doble HL (para que apunten a la siguiente dirección de memoria)

15. Decrementa en una unidad el contenido del registro doble HL (para que apunten a la anterior dirección de memoria)

Instrucciones lógicas

1. Ejecuta la instrucción lógica AND entre el registro B y el acumulador, dejando el resultado en el registro B.

2. Ejecuta la instrucción lógica AND entre el contenido de la posición de memoria 9001h y el acumulador.

3. Queremos saber si MSB del dato guardado en el acumulador es '1'o '0'. Para ello aplicamos una máscara al acumulador para poner a '0' todos los bits menos ese. ¿Cuál es la instrucción lógica que tenemos que ejecutar (indicando el valor)?

4. Ejecuta la instrucción lógica OR entre el registro C y el acumulador, dejando el resultado en el registro C.

5. Ejecuta la instrucción lógica OR entre el contenido de la posición de memoria 900Ah y el acumulador.

6. Queremos poner los bits de peso 0, 1 y 5 del acumulador a '1' sin modificar el resto, para ello aplicamos una máscara al acumulador ¿Cuál es la instrucción lógica que tenemos que ejecutar (indicando el valor)?

7. Ejecuta la instrucción lógica XOR entre el registro D y el acumulador, dejando el resultado en el registro D.

8. Ejecuta la instrucción lógica XOR entre el contenido de la posición de memoria 9000h y el acumulador.

9. Queremos obtener el valor negado del dato contenido en el acumulador, para ello le aplicamos una máscara. ¿Cuál es la instrucción lógica que tenemos que ejecutar (indicando el valor)?

10. Queremos comparar el contenido del registro B y el del Acumulador, sin modificarlos. ¿Cuál es la instrucción lógica que tenemos que ejecutar?, ¿Dónde se guarda el resultado?, ¿Cómo sabemos si A<B, A>B o A=B?

11. ¿Y para compara el contenido del acumulador y el dato en la posición 9000h?

12. ¿Y para comparar el contenido del acumulador con el valor 10_{10} (0Ah)?

Grado en Ingeniería Informática de Gestión y Sistemas de Información
Estructura de Computadores
L6

Instrucciones de rotación

1. Divide entre 2 el número hexadecimal 88h empleando una instrucción de rotación y guarda el resultado en el registro B.

2. Multiplica por 2 el número hexadecimal 11h empleando una instrucción de rotación y guarda el resultado en el acumulador.

3. ¿Cuál es el resultado de ejecutar los programas a) y b) ? Razone la respuesta

a)
```
mvi A, 81
ral
ral
```

b)
```
mvi A, 81
ral
rlc
```

4. ¿Cuál es el resultado de ejecutar los programas A a) y b) ? Razone la respuesta

a)
```
mvi A, 81
rlc
rlc
```

b)
```
mvi A, 81
rlc
ral
```

5. Realiza el complemento a 1 del dato guardado en el registro B, dejando el resultado en el registro B.

6. Realiza el complemento a 2 del dato guardado en el registro C, dejando el resultado en el registro C.

Instrucciones de salto

1. Si el resultado de la operación anterior es cero salta a la etiqueta “cero” si no, salta a la etiqueta “no-cero”

2. Si el resultado de la operación anterior no es cero salta a la etiqueta “no-cero” si no, continúa la ejecución normal del programa

3. Si el resultado de la operación anterior tiene llevada, salta a etiqueta “llevada”, si no, salta a la etiqueta “no llevada”

4. Si el resultado de la operación anterior no tiene llevada, salta a etiqueta “nollevada”, si no, continúa la ejecución normal del programa

5. Si el resultado de la operación anterior tiene un número impar de ‘1’s , salta a etiqueta “impar”, en caso contrario continúa la ejecución normal del programa

6. Si el resultado de la operación anterior tiene un número par de ‘1’s , salta a etiqueta “par”, en caso contrario, continúa la ejecución normal del programa

7. Si hay error en un dato enviado mediante un código de paridad par, ejecuta la subrutina “error” , en caso contrario, continúa la ejecución normal del programa

8. Si el resultado de la operación anterior ha sido positivo, salta a etiqueta “positivo”, en caso contrario, continúa la ejecución normal del programa

9. Si el resultado de la operación anterior ha sido negativo, salta a etiqueta “negativo”, en caso contrario, continúa la ejecución normal del programa

10. Si el dato de la posición de memoria 9000h ha sido negativo, llama a la subrutina “negativo”. Pon un ejemplo.

11. Si el dato de la posición de memoria 9000h tiene un número impar de ‘1’s, introduce el número 01h en el registro B, en caso contrario el número 02h

12. ¿Cuál es la diferencia entre los siguientes programas?:

```
jmp salto1          call salto1
mvi b,01            mvi b,01
salto1:             jmp fin
mvi d,01            salto1:
                    mvi d,01
                    ret
                    fin:
```

13. ¿Cuál es la diferencia entre los siguientes programas?:

```
lda 9000            lda 9000
adi 00              adi 00
jp positivo         cp positivo
mvi b,01            mvi b,01
positivo:           jmp fin
mvi d,01            positivo:
                    mvi d,01
                    ret
                    fin:
```

14. Llama a la subrutina “subrutina”, en ella suma 05h al valor del acumulador y si es positivo regresa a la rutina principal

15. Llama a una subrutina, allí pasa el dato del registro b al acumulador y si el número de ‘1’s es par, vuelve de la subrutina

16. Llama a una subrutina, allí pasa decrementa en una unidad el valor del acumulador y si el resultado es cero vuelve de la subrutina.

Llamadas y E/S

El siguiente código recoge dos números introducidos uno por el puerto 1 y el otro por el puerto 2; los coloca en los registros C y D, y llama a la subrutina “sumarBCD”. ¿Qué hace dicha subrutina?

Escribe el programa y ejecútalo paso a paso para comprobar el funcionamiento. Presta atención a los valores que toman los punteros PC y SP. Rellena la tabla de abajo identificando cada instrucción (ten en cuenta que no todas ocupan lo mismo):

mnemónico	Dirección	Instrucción hex
in 01		
mov c,a		
in 02		
mov d,a		
mvi b,00		
call sumarBCD		
mov a,b		
out 03		
jmp fin		
nop		
nop		
nop		
nop		
nop		
sumarBCD:		
mov a,c		
add d		
mov b,a		
ret		
fin:		

Completos

Escribe y comprueba los códigos para los siguientes programas.

1- En PA se realizó el diagrama de flujo de un programa para multiplicar 2 números y otro que calculaba el factorial (PA3). Escribe y comprueba el código para calcular con el 8085 un factorial.

2- Lee el contenido de la posición 9000, y si es más pequeño que 07h, guarda su valor en la posición 9010.

3- Lee el contenido de la posición 9000h, y si es mayor o igual que 07h, guarda 0Fh en la posición 9010h.

4- En la posición 9000h está el primer elemento de un array. Sabiendo que su longitud es de 10 elementos, escribe un programa que guarda en la posición 9100h la suma de todos sus elementos.

5- En la posición 9000h está el primer elemento de un array. Sabiendo que su longitud es de 10 elementos, escribe un programa que ordene los elementos del array de menor a mayor.

6- En la posición 9000h está el primer elemento de un array. Sabemos que todos los elementos son positivos. De los 10 elementos que lo componen, busca el valor más pequeño y guarda su posición relativa (la que tiene dentro del array) en la posición de memoria siguiente a la del final del array.

7- Escribe un programa que lea indefinidamente el dato de la posición 9001h y si es igual a tres meta dos retardos sin operación para después poner el valor 01h en el puerto 3.

8- Un sensor de nivel pone el byte en la posición 9001h todo a '1's cuando el agua llega a un determinado nivel. Cuando esto ocurre nuestro programa debe deshabilitar las interrupciones y llamar a la subrutina de atención ("atención").