# INSTRUCCIONS DEL I8085

Les instruccions estan descrites en detall i ordenades alfabèticament per facilitar la seva localització. La unitat bàsica de temps és un cicle de rellotge o estat. Típicament una microinstrucció "cicle de màquina" consta de 3 a 6 estats. Les operacions més senzilles només requereixen d'un cicle de màquina. El "cicle de instrucció" és el temps requerit per executar una instrucció completa y pot constar de 1 a 5 cicles de màquina. Les instruccions consten bàsicament de una "fase de recerca" i una "fase de execució". Un estat pot oscil·lar entre els 320 ns i els 2 ns. Sabent la duració d'un estat, a un determinat sistema es pot determinar el temps d'execució d'una instrucció qualsevol, multiplicant el temps de cada estat pel número d'ells que conté cada instrucció. A la descripció de cada una de las instruccions s'especifica:

- El format de la instrucció.
- El codi objecte de la mateixa.
- El número de bytes que ocupa.
- Els cicles necessaris per a la seva execució.
- Els flags afectats per l'execució de la instrucció.
- El mode de direccionament utilitzat.

| INSTRUCCIO | CODI OBJ | BYT. | CIC | FLAGS AFECTATS | DIRECCIONAMIENT |
|---|---|---|---|---|---|
| ACI DATA | CE YY | 2 | 7 | Z,S,P,CY,AC | Inmediato |

Suma el contenido del byte especificado (DATA) en la instrucción, al contenido del acumulador, añadiendo además el bit del acarreo. El resultado se almacena en el acumulador (perdiéndose así el anterior contenido del Acumulador).
El dato (DATA) debe estar especificado en forma de número, en ASCII constante, como etiqueta de un valor previamente definido o una expresión. El dato no debe exceder de un byte.

| | | | | | |
|---|---|---|---|---|---|
| ADC reg | 1000 1XXX | 1 | 4 | Z,S,P,CY,AC | Registro |

Suma el contenido de un registro (reg) con el contenido del acumulador y el bit de acarreo. El resultado queda en el acumulador. El operando (reg) debe especificar uno de los registros del A al E, el H o el L.

| | | | | | |
|---|---|---|---|---|---|
| ADC M | 8E | 1 | 7 | Z,S,P,CY,AC | Registro índice |

Esta instrucción suma el contenido del byte de memoria direccionado por el par de registros H y L con el contenido del acumulador y el bit de acarreo; el resultado es almacenado en el acumulador. (M es una referencia simbólica de los registros H y L).

| | | | | | |
|---|---|---|---|---|---|
| ADD reg | 1000 0XXX | 1 | 4 | Z,S,P,CY,AC | Registro |

Suma el byte de datos contenido en el registro especificado (reg debe ser uno de los registros del A al E, el H o el L) al contenido del acumulador y deja el resultado en el acumulador. (Notar que ADD excluye el flag de acarreo de la suma, pero lo utiliza para indicar el resultado de la operación).

| | | | | | |
|---|---|---|---|---|---|
| ADD M | 86 | 1 | 7 | Z,S,P,CY,AC | Regis indirect |

Suma el contenido de la posición de memoria direccionada por los registros H y L con el contenido del acumulador y deja el resultado en el acumulador. (No suma el acarreo, pero lo usa para completar el resultado de la operación).

| | | | | | |
|---|---|---|---|---|---|
| ADI DATA | C6 YY | 2 | 7 | Z,S,P,CY,AC | Inmediato |

Suma el valor del byte especificado en la instrucción (DATA), al contenido del acumulador y deja el resultado en el acumulador. El dato debe ser expresado en forma de número, un ASCII constante, la etiqueta de un valor previamente definido o una expresión. El dato no debe exceder de un byte.

| | | | | | |
|---|---|---|---|---|---|
| ANA reg | 1010 0XXX | 1 | 4 | Z,S,P,CY,AC | Registro |

El operando (reg) debe especificar uno de los registros del A al E, el H o el L. Esta instrucción realiza una operación lógica Y entre el contenido del registro especificado y el contenido del acumulador, dejando el resultado en el acumulador. El flag de acarreo es puesto a cero. (La operación Y produce un 1 solo cuando los dos bits implicados en la operación son 1).

| | | | | | |
|---|---|---|---|---|---|
| ANA M | A6 | 1 | 7 | Z,S,P,CY,AC | Registr indirec |

Se realiza un producto lógico (operación Y) entre el contenido de la posición de memoria especificada por H y L y el contenido del acumulador, dejando el resultado en el acumulador. El flag de acarreo es puesto a cero.

| | | | | | |
|---|---|---|---|---|---|
| ANI DATA | E6 YY | 2 | 7 | Z,S,P,CY,AC | Inmediato |

Realiza una operación Y lógica entre el dato (DATA) especificado en la instrucción y el contenido del acumulador, el resultado queda en el acumulador. Se pone a cero el flag de acarreo. El dato, que no debe exceder de un byte, puede ser expresado en forma de número, un ASCII constante, la etiqueta de algún valor previamente definido o una expresión.

| INSTRUCCIO | CODI OBJ | BYT. | CIC | FLAGS AFECTATS | DIRECCIONAMIENT |
|---|---|---|---|---|---|
| CALL LABEL | CD PPQQ | 3 | 18 | | Inmed/Reg Indir |

CALL guarda el contenido del contador de programa (la dirección de la próxima instrucción secuencial) dentro del stack y a continuación salta a la dirección especificada por LABEL. Cada instrucción CALL o alguna de sus variantes implica una instrucción RET (retorno), de lo contrario el programa podría "perderse" con consecuencias impredecibles. La dirección debe ser especificada como un número, una etiqueta, o una expresión. La etiqueta es lo más normal (El assembler invierte los bytes alto y bajo de dirección durante el proceso de ensamblado). Las instrucciones CALL se emplean para llamadas a subrutinas y debemos tener presente que siempre emplean el stack.

| | | | | | |
|---|---|---|---|---|---|
| CC LABEL | DC PPQQ | 3 | 9/18 | | Inmed/Reg Indir |

CC comprueba el estado del flag de acarreo. Si el flag está a 1, CC carga el contenido del contador de programa en el stack y a continuación salta a la dirección especificada por LABEL. Si el flag esta a 0, la ejecución del programa continúa con la próxima instrucción de su secuencia normal. Aunque el uso de una etiqueta es lo más normal, la dirección puede ser especificada también como un número o una expresión.

| | | | | | |
|---|---|---|---|---|---|
| CM LABEL | FC PPQQ | 3 | 9/18 | | Inmed/Reg Indir |

CM comprueba el estado del flag del signo. Si el flag esta a 1 (indicando que el contenido del acumulador es negativo) CM manda el contenido del contador de programa al stack y salta a la dirección especificada por LABEL. Si el flag es 0 la ejecución del programa continúa con su secuencia normal. El uso de la etiqueta es lo más corriente, pero la dirección puede especificarse también por un núm o una expresión.

| | | | | | |
|---|---|---|---|---|---|
| CMA | 2F | 1 | 4 | | |

La instrucción CMA complementa en contenido del acumulador. No actúa ningún flag de condición. Esta instrucción no emplea operandos. Para realizar el complemento a dos, se suma 1 al contenido del acumulador después de haber ejecutado la instrucción CMA.

| | | | | | |
|---|---|---|---|---|---|
| CMC | 3F | 1 | 4 | CY | |

Complementa el flag de acarreo. Si dicho flag está a 0, CMC lo pone a 1 y si esta a 1 lo pone a 0. El resto de los flags no varían.

| | | | | | |
|---|---|---|---|---|---|
| CMP reg | 1011 1XXX | 1 | 4 | Z,S,P,CY,AC | Registro |

El operando (reg) debe nombrar uno de los registros del A al E, el H o el L. La instrucción CMP reg compara el byte contenido en el registro especificado con el contenido del acumulador. Se indica el resultado actuando los flags de cero y acarreo. Los valores que están siendo comparados permanecen invariables.

El flag de cero indica igualdad. Si hay un 0 en el flag de acarreo, indica que el contenido del acumulador es mayor que el contenido del registro especificado. Un 1 en el acarreo, indica lo contrario. Sin embargo el significado del flag de acarreo es invertido cuando los valores tienen signos diferentes o cuando uno de los valores es complementado.

| | | | | | |
|---|---|---|---|---|---|
| CMP M | BE | 1 | 7 | Z,S,P,CY,AC | Registro |

La instrucción compara el contenido de la posición de memoria direccionada por los registros H y L con el contenido del acumulador. M es una referencia simbólica al par de registros HL. La actuación de los flags es igual a la producida en el caso anterior.

| | | | | | |
|---|---|---|---|---|---|
| CNC LABEL | D4 PPQQ | 3 | 9/18 | | Inmedi/Reg indi |

CNC chequea el valor del flag de acarreo. Si está en cero CNC carga el contenido de contador de programa en el stack y a continuación salta a la dirección especificada por la instrucción en LABEL. Si el flag está a 1, el programa continúa con su secuencia normal. Aunque el uso de una etiqueta es lo más común, la dirección puede también estar indicada por un número o por una expresión.

| | | | | | |
|---|---|---|---|---|---|
| CNZ LABEL | C4 PPQQ | 3 | 9/18 | | Inmed/Reg Indir |

CNZ chequea el flag de Cero. Si está en 0 (indicando que el contenido del acumulador no es cero), CNZ manda el contenido del contador de programa al stack y salta a la dirección especificada por LABEL. Si el flag está a 1 el programa continúa su desarrollo normal.

| | | | | | |
|---|---|---|---|---|---|
| CP LABEL | F4 PPQQ | 3 | 9/18 | | Inmed/Reg Indir |

CP chequea el valor del flag de signo. Si está a 0 (indicando que el contenido del acumulador es positivo), CP envía el contenido del contador de programa al stack y salta a la dirección especificada por LABEL. Si el flag tiene un 1, continúa el programa normalmente con la instrucción siguiente.

| | | | | | |
|---|---|---|---|---|---|
| CPE LABEL | EC PPQQ | 3 | 9/18 | | Inmedi/Reg Indi |

Existe paridad en un byte si el número de unos que tiene es par. El flag de paridad se pone a 1 para indicar esta condición. CPE chequea el valor del flag de paridad. Si tiene un 1, CPE envía el contenido del contador de programa al stack y salta a la dirección especificada por la instrucción en LABEL. Si el flag tiene un cero, el programa continúa normalmente.

| INSTRUCCIO | CODI OBJ | BYT. | CIC | FLAGS AFECTATS | DIRECCIONAMIENT |
|---|---|---|---|---|---|
| CPI DATA | FE YY | 2 | 7 | Z,S,P,CY,AC | Reg Indirecto |

Compara el valor del byte especificado (DATA) con el contenido del acumulador y posiciona los flags de cero y acarreo para indicar el resultado. El flag de cero indica igualdad. Un 0 en el acarreo indica que el contenido del acumulador es mayor que DATA. Un 1 en el acarreo indica que el acumulador es menor que DATA. Sin embargo, el significado del flag de acarreo es contrario cuando los valores tienen diferente signo o cuando uno de los valores está complementado. El valor de DATA no debe exceder de un byte.

| | | | | | |
|---|---|---|---|---|---|
| CPO LABEL | E4 PPQQ | 3 | 9/18 | | Inmedi/Reg Indi |

CPO chequea el flag de paridad. Si el flag esta a 0, CPO carga el contenido del contador de programa en el stack y salta a la dirección especificada en LABEL. Si el flag está a 1 el programa continúa su desarrollo normal.

| | | | | | |
|---|---|---|---|---|---|
| CZ LABEL | CC PPQQ | 3 | 9/18 | | Inmedi/Reg Indi |

CZ chequea el flag de cero. Si el flag esta a 1 (indicando que el contenido del acumulador es cero), CZ carga el contenido del contador de programa en el stack y salta a la dirección especificada en LABEL. Si el flag está a 0 (indicando que el contenido del acumulador es distinto de cero) el programa continúa su desarrollo normal.

| | | | | | |
|---|---|---|---|---|---|
| DAA | 27 | 1 | 4 | Z,S,P,CY,AC | Registro |

La instrucción DAA ajusta el valor de los ocho bits del acumulador para formar dos grupos de cuatro bits binarios codificados en decimal. Esta instrucción no tiene operandos. DAA se emplea cuando deseamos trabajar con números decimales. Es la única instrucción cuya función requiere el uso del flag de acarreo auxiliar. En operaciones aritméticas multi-byte, la instrucción DAA es codificada inmediatamente después de la instrucción aritmética, de tal manera que el flag de acarreo auxiliar no se altera involuntariamente. DAA opera como sigue: 1.- Si los cuatro bits menos significativos del acumulador tienen un valor mayor que 9, o si el flag de acarreo auxiliar está en 1, DAA suma seis al acumulador. 2.- Si los cuatro bits más significativos del acumulador tienen un valor superior a 9, o si el flag de acarreo esta a 1, DAA suma 6 a los cuatro bits más significativos del acumulador.

| | | | | | |
|---|---|---|---|---|---|
| DAD RP | 00XX 1001 | 1 | 10 | CY | Registro |

DAD RP suma el valor de un dato de 16 bits contenido en un determinado par de registros (RP) al contenido del par de registros HL. El resultado es almacenado en el par HL. Los operandos (RP) pueden ser B = BC, D = DE, H = HL, SP. Téngase en cuenta que la letra H debe ser empleada para especificar que el par de registros HL debe ser doblado. DAD pone el flag de acarreo a 1 si hay una salida de acarreo de los registros HL. Y además no afecta a ningún otro flag.

| | | | | | |
|---|---|---|---|---|---|
| DCR reg | 00XX X101 | 1 | 4 | Z,S,P,AC | Registro |

El operando (reg) debe especificar uno de los registros del A al E, el H o el L. La instrucción resta 1 del contenido del registro especificado. Afecta a todos los flags excepto al de acarreo. (Puede usarse dentro de rutinas aritméticas multi-byte para decrementar contadores característicos y propósitos similares).

| | | | | | |
|---|---|---|---|---|---|
| DCR M | 35 | 1 | 10 | Z,S,P,AC | Registro Indire |

Esta instrucción resta 1 del contenido de la posición de memoria direccionada por el par de registros HL.

| | | | | | |
|---|---|---|---|---|---|
| DCX RP | 00XX 1011 | 1 | 6 | | Registro |

DCX resta 1 al contenido del par de registro especificado (RP). Dado que DCX no usa ningún flag, puede emplearse para modificar direcciones en cualquier secuencia de instrucciones dada la estaticidad de los flags. (La letra H puede emplearse para especificar el par de registros HL).

| | | | | | |
|---|---|---|---|---|---|
| DI | F3 | 1 | 4 | | |

Después de la ejecución de una instrucción DI, el sistema de "interrupciones" esta sin posibilidad de ponerse en marcha. En aplicaciones que empleen las interrupciones, la instrucción DI se emplea solamente cuando una determinada secuencia no debe ser interrumpida. Por ejemplo, se puede poner fuera de servicio el sistema de interrupciones incluyendo una instrucción DI el principio del código de secuencia. La interrupción TRAP del 8085 no puede ser puesta fuera de servicio. Esta interrupción especial esta prevista para serios problemas que pueden presentarse independientemente del flag de interrupción (fallo de alimentación, etc.).

| | | | | | |
|---|---|---|---|---|---|
| EI | FB | 1 | 4 | | |

La instrucción EI pone en servicio el sistema de interrupciones a partir de la siguiente instrucción secuencial del programa.

Se puede desconectar el sistema de interrupciones poniendo una instrucción DI al principio de una secuencia, puesto que no se puede predecir la llegada de una interrupción. Al final de la secuencia se incluye la instrucción EI que vuelve a habilitar el sistema de interrupciones. (RESET también pone fuera de servicio el sistema de interrupciones además de poner el contador de programa a cero).

| INSTRUCCIO | CODI OBJ | BYT. | CIC | FLAGS AFECTATS | DIRECCIONAMIENT |
|---|---|---|---|---|---|
| HLT | 76 | 1 | 4 | | |

La instrucción HLT detiene el procesador. El contador de programa contiene la dirección de la próxima instrucción secuencial. Por otro lado los flags y registros permanecen inactivos. Una vez en estado de parada el procesador puede volver a ser arrancado solamente por un acontecimiento externo, es decir una interrupción. Por tanto debemos asegurarnos que las interrupciones estén en disposición de ser activadas antes de ejecutar la instrucción HLT. Si se ejecuta HLT estando las interrupciones fuera de servicio, la única manera de volver arrancar el procesador será mediante un RESET o a través de la interrupción TRAP. El procesador puede salir temporalmente del estado de parada para servir un acceso directo a memoria, sin embargo terminado el acceso directo vuelve al estado de parada.
Un propósito básico de la instrucción HLT es permitir una pausa al procesador mientras espera por la interrupción de un periférico.

| | | | | | |
|---|---|---|---|---|---|
| IN PORT | DB YY | 2 | 10 | | Directo |

La instrucción IN PORT lee los 8 bits de datos que hay en el "PORT" especificado y los carga en el acumulador. El operando debe ser un número o una expresión que produzca un valor comprendido entre 00H y FFH.

| | | | | | |
|---|---|---|---|---|---|
| INR reg | 00XX X100 | 1 | 4 | Z,S,P,AC | Registro |

El operando (reg) debe especificar uno de los registros del A al E, el H o el L. La instrucción suma 1 al contenido del registro especificado.

| | | | | | |
|---|---|---|---|---|---|
| INR M | 34 | 1 | 10 | Z,S,P,AC | Registro Indire |

La instrucción suma 1 al contenido de la dirección de memoria señalada por el par de registros HL. M es una referencia simbólica a los registros H y L.

| | | | | | |
|---|---|---|---|---|---|
| INX RP | 00XX 0011 | 1 | 6 | | Registro |

La instrucción INX suma 1 al par de registros especificado. No afecta a ningún flag. Puede usarse para incrementar los pares de registros BC, DE, HL y SP. (Cuidado al incrementar SP).

| | | | | | |
|---|---|---|---|---|---|
| JC LABEL | DA PPQQ | 3 | 7/10 | | Inmediato |

La instrucción JC LABEL comprueba el valor del flag de acarreo. Si es un 1 la ejecución del programa continúa en la dirección especificada por LABEL. Si es un 0 el programa continúa su ejecución normal de forma secuencial.

| | | | | | |
|---|---|---|---|---|---|
| JM LABEL | FA PPQQ | 3 | 7/10 | | Inmediato |

La instrucción JM LABEL comprueba el estado del flag de signo. Si el contenido del acumulador es negativo (flag de signo = 1) la ejecución del programa continúa en la dirección especificada por LABEL. Si el contenido del acumulador es positivo (flag de signo = 0) continúa la ejecución de la secuencia normal.

| | | | | | |
|---|---|---|---|---|---|
| JMP LABEL | C3 PPQQ | 3 | 10 | | Inmediato |

La instrucción JMP LABEL altera la ejecución del programa cargando el valor especificado por LABEL en el contador de programa.

| | | | | | |
|---|---|---|---|---|---|
| JNC LABEL | D2 PPQQ | 3 | 7/10 | | Inmediato |

La instrucción JNC LABEL comprueba el estado del flag acarreo. Si esta a 0 el programa cambia a la dirección especificada por LABEL. Si esta a 1 la ejecución del programa continúa normalmente.

| | | | | | |
|---|---|---|---|---|---|
| JNZ LABEL | C2 PPQQ | 3 | 7/10 | | Inmediato |

La instrucción JNZ LABEL comprueba el valor del flag de cero. Si el contenido del acumulador no es cero (Flag de cero = 0) el programa continúa en la dirección especificada por LABEL. Si el contenido del acumulador es cero (Flag de cero = 1) el programa continúa su ciclo normal.

| | | | | | |
|---|---|---|---|---|---|
| JP LABEL | F2 PPQQ | 3 | 7/10 | | Inmediato |

La instrucción JP LABEL comprueba el estado de flag del signo. Si el contenido del acumulador es positivo (flag de signo = 0) la ejecución del programa continúa con la dirección especificada por LABEL. Si el contenido del acumulador es negativo (flag signo=1) continúa el programa con su ejecución normal.

| | | | | | |
|---|---|---|---|---|---|
| JPE LABEL | EA PPQQ | 3 | 7/10 | | Inmediato |

La paridad existe si el byte que esta en el acumulador tiene un número par de bits. El flag de paridad se pone a 1 para indicar esta condición.
La instrucción JPE LABEL comprueba la situación del flag de paridad. Si esta a 1, la ejecución del programa continúa en la dirección especificada por LABEL. Si esta a 0, continúa con la siguiente instrucción de forma secuencial.
Las instrucciones JPE y JPO son especialmente usadas para comprobar la paridad de los datos de entrada. (Sin embargo con la instrucción IN los flags no actúan. Esto puede evitarse sumando 00H al acumulador para activarlos).

| | | | | | |
|---|---|---|---|---|---|
| JPO LABEL | E2 PPQQ | 3 | 7/10 | | Inmediato |

La instrucción JPO LABEL comprueba el estado del flag de paridad. Si esta a 0, el programa continúa en la dirección marcada por LABEL. Si está a 1 continúa con la secuencia normal.

| | | | | | |
|---|---|---|---|---|---|
| JZ LABEL | CA PPQQ | 3 | 7/10 | | Inmediato |

La instrucción JZ LABEL comprueba el flag de cero. Si está a 1 el programa continúa en la dirección expresada por LABEL. Si está a 0 continúa con la ejecución secuencial normal.

| | | | | | |
|---|---|---|---|---|---|
| LDA ADDR | 3A PPQQ | 3 | 13 | | Directo |

LDA ADDR carga el acumulador con el contenido de la memoria direccionada por ADDR. La dirección puede ser puesta como un número, una etiqueta previamente definida o una expresión.

| | | | | | |
|---|---|---|---|---|---|
| LDAX RP | 000X 1010 | 1 | 7 | | Registro indire |

LDAX RP carga el acumulador con una copia del byte almacenado en la localización de memoria direccionada por el par de registros BC o DE. (El par BC se especifica con B y el par DE con D).

| | | | | | |
|---|---|---|---|---|---|
| LHLD ADDR | 2A PPQQ | 3 | 16 | | Directo |

LHLD ADDR carga el registro L con una copia del byte almacenado en la posición de memoria especificada por ADDR. Después carga el registro H con una copia del byte almacenado en la posición siguiente de memoria especificada por ADDR. La instrucción LHLD esta prevista para cargar direcciones nuevas en los registros H y L.

| | | | | | |
|---|---|---|---|---|---|
| LXI RP,DA16 | 00XX 0001 YY YY | 3 | 10 | | Inmediato |

LXI es una instrucción de 3 bytes; su segundo y tercer byte contienen el dato que ha de ser cargado en el par de registros (RP). El primer operando debe especificar el par de registros a ser cargados, pueden ser los pares BC, DE, HL, o el SP. El segundo operando especifica los dos bytes a ser cargados. LXI es la única instrucción inmediata que acepta un valor de 16 bits. El resto trabajan con datos de 8 bits.

| | | | | | |
|---|---|---|---|---|---|
| MOV reg,reg | 01dddsss | 1 | 4 | | Registro |

Copia el contenido del segundo registro en el primero. Cada operando debe especificar uno de los registros A, B, C, D, E, H o L.

| | | | | | |
|---|---|---|---|---|---|
| MOV M,reg | 0111 0sss | 1 | 7 | | Registro Indire |

Copia el contenido del registro especificado en la posición de memoria direccionada por los registros H y L. El segundo operando (reg) debe especificar uno de los registros A, B, C, D, E, H o L.

| | | | | | |
|---|---|---|---|---|---|
| MOV reg,M | 01dd d110 | 1 | 7 | | Reggist.Indirec |

Copia el contenido de la posición de memoria direccionada por los registros H y L en el registro especificado. El primer operando debe especificar el registro deseado como destino.

| | | | | | |
|---|---|---|---|---|---|
| MVI reg,DAT | 00ddd110 YY | 2 | 7 | | Inmediato |

El primer operando debe ser uno de los registros A,B,C,D,E,H o L, que será cargado con el dato especificado en el segundo operando (DATA). El dato no debe exceder de un byte.

| | | | | | |
|---|---|---|---|---|---|
| MVI M,DATA | 36 YY | 2 | 10 | | Inmedi/Reg Indi |

Esta instrucción carga el dato especificado, DATA, a la posición de memoria direccionada por el par HL.

| | | | | | |
|---|---|---|---|---|---|
| NOP | 00 | 1 | 4 | | |

NOP no realiza ninguna operación y no afecta a ninguno de los flags de condición. Se emplea normalmente para completar ciclos de lazo.

| | | | | | |
|---|---|---|---|---|---|
| ORA reg | 1011 0XXX | 1 | 4 | Z,S,P,CY,AC | Registro |

El operando debe especificar uno de los registros del A al E, el H o el L. Esta instrucción realiza una operación lógica "O" entre el contenido del registro especificado y el acumulador, dejando el resultado en el acumulador. Los flags de acarreo y acarreo auxiliar se ponen a cero.

| | | | | | |
|---|---|---|---|---|---|
| ORA M | B6 | 1 | 7 | Z,S,P,CY,AC | Reg Indirecto |

Realiza una operación "O" entre el contenido de la dirección de memoria especificada por el par HL y el contenido del acumulador. El resultado es almacenado en el acumulador. Los flags de acarreo y acarreo auxiliar son puestos a cero.

| | | | | | |
|---|---|---|---|---|---|
| ORI DATA | F6 YY | 2 | 7 | Z,S,P,CY,AC | Inmediato |

ORI desarrolla una operación lógica "O inclusiva" entre el contenido especificado por DATA y el contenido del acumulador. El resultado se deja en el acumulador. Los flags de acarreo y acarreo auxiliar se ponen a cero.

| | | | | | |
|---|---|---|---|---|---|
| OUT PORT | D3 YY | 2 | 10 | | Directo |

OUT PORT pone el contenido del acumulador en el bus de datos de 8 bits del puerto seleccionado en el bus de direcciones de 16 bits. El número de puertos oscila de 0 a 255 y es duplicado en el bus de direcciones. Es la lógica externa la encargada de seleccionar el puerto y aceptar el dato de salida. El operando (PORT) debe especificar el número del puerto de salida seleccionado.

| | | | | | |
|---|---|---|---|---|---|
| PCHL | E9 | 1 | 6 | | Registro |

PCHL carga el contenido de los registros H y L en el contador de programa. Como el procesador busca la siguiente instrucción en la siguiente dirección del contador de programa, PCHL tiene el efecto de una instrucción de salto. El contenido de H va a los 8 bits más altos de contador de programa y el contenido de L va a los 8 bits más bajos.

| | | | | | |
|---|---|---|---|---|---|
| POP RP | 11XX 0001 | 1 | 10 | | Reg. Indirecto |

POP RP copia el contenido de la posición de memoria direccionada por el stack pointer en el registro de bajo orden del par de registros especificados. A continuación se incrementa el stack pointer en 1 y copia el contenido de la dirección resultante en el registro de más alto orden del par. Luego se incrementa el stack pointer otra vez de modo que se apunta al siguiente dato del stack. El operando debe especificar el par de registros BC, DE, HL o PSW.

POP PSW usa el contenido de la localización de memoria especificada por el stack pointer para restablecer los flags de condiciones.

| | | | | | |
|---|---|---|---|---|---|
| PUSH RP | 11XX 0101 | 1 | 12 | | Reg.Indirecto |

PUSH copia dos bytes de datos en el stack. El dato puede ser el contenido de un par de registros o la "palabra de estado del programa". PUSH resta 1 del stack pointer y copia el contenido del registro de "alto orden" del par de registros en la dirección resultante. A continuación se resta otra vez 1 al stack pointer y se copia el registro de bajo orden en la dirección resultante. Los registros permanecen invariables.

| | | | | |
|---|---|---|---|---|
| RAL | 17 | 1 | 4 | CY |

RAL hace girar el contenido del acumulador y el flag de acarreo un espacio de un bit hacia la salida (izquierda). El flag de acarreo que es tratado como si fuera del acumulador, se transfiere el bit de menor orden del acumulador. El bit de mayor orden del acumulador se transfiere al flag de acarreo. No tiene operandos.

| | | | | |
|---|---|---|---|---|
| RAR | 1F | 1 | 4 | CY |

RAR rota el contenido del acumulador y del flag de acarreo 1 bit de posición a la derecha. El flag de acarreo que es tratado como si fuera parte del acumulador se transfiere al bit de más alto orden del acumulador. El bit de menor peso del acumulador se carga en el flag de acarreo. No existen operandos en la instrucción RAR.

| | | | | |
|---|---|---|---|---|
| RC | D8 | 1 | 6/12 | Reg. Indirecto |

La instrucción RC comprueba el estado del flag de acarreo. Si tiene un 1 (indicando que hay acarreo) la instrucción saca dos bytes del stack y los mete en el contador de programa. El programa continúa en la nueva dirección suministrada. Si el flag es 0, el programa continúa en la siguiente instrucción de la secuencia normal.

| | | | | |
|---|---|---|---|---|
| RET | C9 | 1 | 10 | Reg. Indirecto |

La instrucción RET echa fuera dos bytes de datos del stack y los mete en el registro contador de programa. El programa continúa entonces en la nueva dirección. Normalmente RET se emplea conjuntamente con CALL.

| | | | |
|---|---|---|---|
| RIM | 20 | 1 | 4 |

RIM carga los 8 bits de datos siguientes en el acumulador:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SID | I7.5 | I6.5 | I5.5 | IE | M7.5 | M6.5 | M5.5 |

SID = Bit presente en la entrada serie
I7.5 = Interrupción 7.5 pendiente si esta a 1
I6.5 = Interrupción 6.5 pendiente si esta a 1
I5.5 = Interrupción 5.5 pendiente si esta a 1
IE = Las interrupciones son autorizadas si es 1
M7.5 = La interrupción 7.5 está prohibida si está a 1
M6.5 = La interrupción 6.5 está prohibida si está a 1
M5.5 = La interrupción 5.5 está prohibida si está a 1

| | | | | |
|---|---|---|---|---|
| RLC | 07 | 1 | 4 | CY |

RLC rota un bit hacia la izquierda todo el contenido del acumulador, transfiriendo el bit de más alto orden al flag de acarreo y al mismo tiempo a la posición de menor orden del acumulador.

| | | | | |
|---|---|---|---|---|
| RM | F8 | 1 | 6/12 | Reg. Indirecto |

La instrucción RM comprueba el flag de signo. Si tiene un 1, indicando dato negativo en el acumulador, la instrucción echa dos bytes fuera del stack y los mete en el contador de programa. Si el flag tiene 0, continúa el programa normal con la siguiente instrucción.

| | | | | | |
|---|---|---|---|---|---|
| RNC | D0 | 1 | 6/12 | | Reg. Indirecto |

La instrucción RNC comprueba el flag de acarreo. Si está a 0 indicando que no hay acarreo, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el flag está a 1 continúa el ciclo normal.

| | | | | | |
|---|---|---|---|---|---|
| RNZ | C0 | 1 | 6/12 | | Reg. Indirecto |

La instrucción RNZ comprueba el flag cero. Si está a 0, indicando que el contenido del acumulador no es cero, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el flag está a 1, continúa el ciclo normal.

| | | | | | |
|---|---|---|---|---|---|
| RP | F0 | 1 | 6/12 | | Reg. Indirecto |

La instrucción RP comprueba el flag signo. Si está a 0, indicando que el contenido del acumulador es positivo, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el flag está a 1 continúa el ciclo normal.

| | | | | | |
|---|---|---|---|---|---|
| RPE | E8 | 1 | 6/12 | | Reg. Indirecto |

La instrucción RPE comprueba el flag de paridad. Si está a 1, indicando que existe paridad, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el flag está a 0 continúa el ciclo normal. (Existe paridad si el byte que está en el acumulador tiene un número par de bits, colocándose el flag de paridad a 1 en este caso).

| | | | | | |
|---|---|---|---|---|---|
| RPO | E0 | 1 | 6/12 | | Reg. Indirecto |

La instrucción RPO comprueba el flag de paridad. Si está a 0, indicando que no hay paridad, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el flag está a 1, continúa el ciclo normal.

| | | | | | |
|---|---|---|---|---|---|
| RRC | 0F | 1 | 4 | CY | |

RRC rota el contenido del acumulador un bit a la derecha, transfiriendo el bit de más bajo orden a la posición de más alto orden del acumulador, además pone el flag de acarreo igual al bit de menor orden del acumulador.

| | | | | | |
|---|---|---|---|---|---|
| RST N | 11XX X111 | 1 | 12 | | Reg. Indirecto |

Es una instrucción CALL para usar con interrupciones. RST carga el contenido del contador de programa en el stack, para proveerse de una dirección de retorno y salta a una de las "ocho" direcciones determinadas previamente. Un código de tres bits incluido en el código de operación de la instrucción RST especifica la dirección de salto. Esta instrucción es empleada por los periféricos cuando intentan una interrupción.

La instrucción RST tiene el siguiente formato:

| 1 1 C C C 1 1 1 |
|---|

Luego según la combinación de 0 y 1 que demos a C C C obtendremos los distintos formatos y las distintas direcciones de las interrupciones, que serán:

| FORMATO | RST | CONTADOR DE PROGRAMA | |
|---|---|---|---|
| 1100 0111 | C7 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0000H |
| 1100 1111 | CF | 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 | 0008H |
| 1101 0111 | D7 | 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 | 0010H |
| 1101 1111 | DF | 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 | 0018H |
| 1110 0111 | E7 | 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 | 0020H |
| 1110 1111 | EF | 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 | 0028H |
| 1111 0111 | F7 | 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 | 0030H |
| 1111 1111 | FF | 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 | 0038H |

| | | | | | |
|---|---|---|---|---|---|
| RZ | C8 | 1 | 6/12 | | Reg.Indirecto |

La instrucción RZ comprueba el flag de cero. Si está a 1, indicando que el contenido del acumulador es cero, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el flag está a 0, continúa el ciclo normal.

| | | | | | |
|---|---|---|---|---|---|
| SBB reg | 1001 1XXX | 1 | 4 | Z,S,P,CY,AC | Registro |

SBB reg resta uno de los registros del A al E, el H o el L y el flag de acarreo, del contenido del acumulador, dejando el resultado en el acumulador.

| | | | | | |
|---|---|---|---|---|---|
| SBB M | 9E | 1 | 7 | Z,S,P,CY,AC | Reg. Indirecto |

Esta instrucción resta el flag de acarreo y el contenido de la posición de memoria direccionada por los registros H y L, del contenido del acumulador y deja el resultado en el acumulador.

| INSTRUCCIO | CODI OBJ | BYT. | CIC | FLAGS AFECTATS | DIRECCIONAMIENT |
|---|---|---|---|---|---|
| SBI DATA | DE YY | 2 | 7 | Z,S,P,CY,AC | Inmediato |

SBI resta el contenido de DATA y el flag de acarreo, del contenido del acumulador, dejando el resultado en el acumulador.

| | | | | | |
|---|---|---|---|---|---|
| SHLD ADDR | 22 PPQQ | 3 | 16 | | Directo |

SHLD almacena una copia del registro L en la posición de memoria especificada por ADDR, a continuación almacena una copia del registro H en la siguiente posición de memoria (ADDR+1). SHLD es una instrucción proyectada para salvar el contenido del par HL.

| | | | | | |
|---|---|---|---|---|---|
| SIM | 30 | 1 | 4 | | |

SIM es una instrucción de usos múltiples que utiliza el contenido del acumulador para posicionar el enmascaramiento de interrupciones para las RST 5.5, RST 6.5, RST 7.5; pone a cero el flanco sensitivo de la entrada RST 7.5 y saca el bit 7 del acumulador al latch de datos de salida serie.
La estructura de la instrucción SIM es como sigue:

SOD SOE X R7.5 MSE M7.5 M6.5 M5.5

SOD = Bit a presentar sobre la salida serie.
SOE = La salida serie está autorizada si esta a 1.
R7.5 = Reset de RST 7.5.Si es 1 el flip-flop se pone a 0
MSE = Si es un 1 los enmascarados están autorizados.
M7.5 = Si es 1 la interrupción 7.5 queda prohibida.
M6.5 = Si es 1 la interrupción 6.5 queda prohibida.
M5.5 = Si es 1 la interrupción 5.5 queda prohibida.

Si el bit 3 se pone a 1, la función poner "mask" se pone enable (permitida). Los bits 0 al 2 ponen en servicio la correspondiente interrupción RST colocando un 0 en la interrupción que deseamos habilitar. Si colocamos un 1 en alguno de los bits 0 al 2, la interrupción correspondiente no se cumplirá. Si el bit 3 tiene un 0, los bits 0 al 2 no tienen efectos. Se debe usar esta peculiaridad para enviar un bit de salida serie sin afectar al enmascaramiento de las interrupciones. (La instrucción DI anula la SIM).
Si el bit 6 (SOE) se pone a 1 se habilita la función de salida serie. El bit 7 se sitúa en la salida SOD donde puede ser tratado por los aparatos periféricos. Si el bit 6 se pone a cero, el bit 7 no tendrá efecto alguno, siendo ignorado.

| | | | | | |
|---|---|---|---|---|---|
| SPHL | F9 | 1 | 6 | | |

SPHL carga el contenido de los registros H y L en el stack.

| | | | | | |
|---|---|---|---|---|---|
| STA ADDR | 32 PPQQ | 3 | 13 | | Directo |

STA ADDR almacena una copia del contenido actual del acumulador en la posición de memoria especificada por ADDR.

| | | | | | |
|---|---|---|---|---|---|
| STAX RP | 000X 0010 | 1 | 7 | | Reg. Indirecto |

STAX RP almacena una copia del contenido del acumulador en la posición de memoria direccionada por el par de registros especificados por RP (Par BC o par DE).

| | | | | | |
|---|---|---|---|---|---|
| STC | 37 | 1 | 4 | CY | |

STC pone el flag de acarreo a 1. No afecta a otro flag.

| | | | | | |
|---|---|---|---|---|---|
| SUB reg | 001 0xxx | 1 | 4 | Z,S,P,CY,AC | Registro |

El operando debe especificar uno de los registros del A al E, el H o el L. La instrucción resta el contenido del registro especificado del contenido del acumulador, usando representación de los datos en complemento a dos. El resultado es almacenado en el acumulador.

| | | | | | |
|---|---|---|---|---|---|
| SUB M | 96 | 1 | 7 | Z,S,P,CY,AC | Reg. Indirecto |

La instrucción resta el contenido de la posición de memoria direccionada por los registros H y L del contenido del acumulador. El resultado es almacenado en el acumulador.

| | | | | | |
|---|---|---|---|---|---|
| SUI DATA | D6 YY | 2 | 7 | Z,S,P,CY.AC | Inmediato |

SUI DATA resta el contenido de DATA del contenido del acumulador y almacena el resultado en el acumulador. La instrucción SUI utiliza el flag de acarreo durante la sustracción, pero acciona dicho flag para indicar la salida de la operación.

| | | | | | |
|---|---|---|---|---|---|
| XCHG | EB | 1 | 4 | | Registro |

XCHG cambia el contenido de los registros H y L con el contenido de los registros D y E.

| | | | | | |
|---|---|---|---|---|---|
| XRA reg | 010 1XXX | 1 | 4 | Z,S,P,CY,AC | Registro |

Esta instrucción realiza una "O exclusiva" usando el contenido del registro especificado y el contenido del acumulador. El resultado se almacena en el acumulador. Los dos flags de acarreo se ponen a 0. (Una O exclusiva produce un 1 cuando los dos datos son diferentes).

| INSTRUCCIO | CODI OBJ | BYT. | CIC | FLAGS AFECTATS | DIRECCIONAMIENT |
|---|---|---|---|---|---|
| XRA M | AE | 1 | 7 | Z,S,P,CY,AC | Reg. Indirecto |

Se realiza una O exclusiva entre el contenido de la posición de memoria especificada por HL y el contenido del acumulador, quedando el resultado en éste. Los dos flags de acarreo se ponen a 0. (La instrucción XRA A pone a cero el acumulador).

| | | | | | |
|---|---|---|---|---|---|
| XRI DATA | EE YY | 2 | 7 | Z,S,P,CY,AC | Inmediato |

XRI DATA realiza una O exc lusiva entre el contenido de DATA y el contenido del acumulador. El resultado se pone en el acumulador. Pone los flags de acarreo a cero.

| | | | | | |
|---|---|---|---|---|---|
| XTHL | E3 | 1 | 16 | | Reg. Indirecto |

XTHL cambia los dos bytes de la posición más alta del stack con los dos bytes almacenados en los registros H y L. Así XTHL salva el contenido actual del par HL y carga nuevos valores en HL.
XTHL cambia el contenido del L con la posición de memoria especificada por el stack pointer y el registro H es intercambiado con el contenido del SP+1.

## SET D'INSTRUCCIONS DEL I8085

Transferència de dades

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **mov** | **mvi** | **ldax** | **lhld** | **lda** | **stax** | **shld** | **sta** |
| **lxi** | **sphl** | **xthl** | **xchg** | **out** | **in** | **push** | **pop** |

Artitmètiques

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **add** | **adc** | **adi** | **aci** | **sub** | **sbb** | **sui** | **sbi** |
| **dad** | **daa** | **inr** | **enx** | **dcr** | **dcx** | | |

Loguiqes

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **stc** | **ana** | **ani** | **xra** | **xri** | **ora** | **ori** | **cmp** |
| **cpi** | **rlc** | **rrc** | **ral** | **rar** | **cma** | **cmc** | |

Salt

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **jmp** | **jnz** | **jz** | **jnc** | **jc** | **jpo** | **jpe** | **jp** |
| **jm** | **pchl** | **rst** | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **call** | **cnz** | **cz** | **cnc** | **cc** | **cpo** | **cpe** | **cp** |
| **cm** | | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **ret** | **rnz** | **rz** | **rnc** | **rc** | **rpo** | **rpe** | **rp** |
| **rm** | | | | | | | |

Control

| | | | | | |
|---|---|---|---|---|---|
| **nop** | **hlt** | **di** | **ei** | **rim** | **sim** |