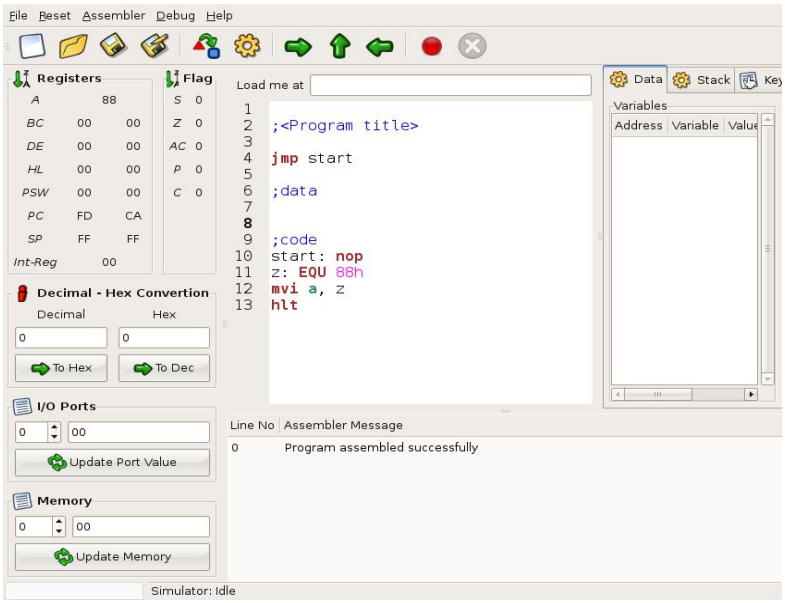
Traducción del manual al español

GNUSIM8085

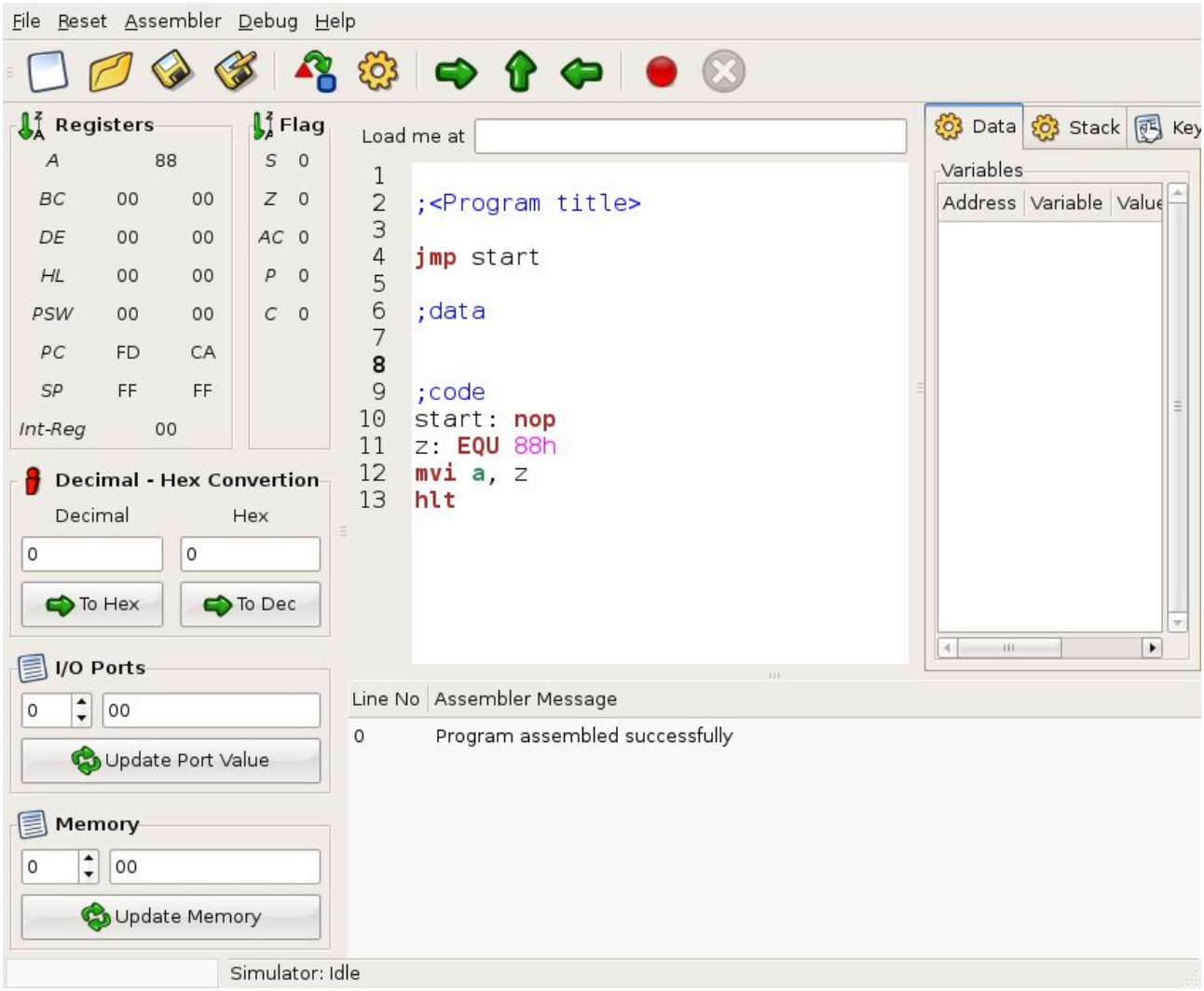


Charles Escobar

2008

Índice de contenido

GNUSim8085



GNUSIM8085 simula el comportamiento del procesador Intel (r) 8085, el cual es la base de los procesadores de la familia ix86.

Esta herramienta permite cargar y ejecutar programas realizados con el conjunto de instrucciones del procesador i8085 para estudiar:

- El contenido de los registros A, B, C, D, E, H, L, PSW, PC, SP, Int-Reg y BANDERAS
- Entrada/Salida de los puertos
- Comportamiento de la memoria
- Asignación de variables y campos de memoria
- Comportamiento de la pila (STACK)

Adicionalmente posee un conversor entre los sistemas numéricos binario y hexadecimal

Manual de Referencia Ensamblador GNUSim8085

Author: Sridhar Ratna <srid@nearfar.org>

1. Introducción

Un programa básico en assembler GNUSIM8085 consiste de 4 partes:

a. Operaciones de máquina (mnemónicos)
b. Pseudo operaciones (Como preprocesador en C)
c. Etiquetas
d. Comentarios

Además, puede contener constantes. A menos que se especifique lo contrario, una constante, la cual es siempre numérica, estará expresada en forma decimal.

Si se adiciona el caracter 'h' se asumirá en forma hxadecimal. Si una constante hexadecimal comienza con un caracter alfabético, no olvide incluir el '0' al inicio, esto ayudará al assembler a diferenciar entre una etiqueta y una constante.

2. Etiquetas

Las etiquetas sirven para identificar a cualquier instruction/dato particular en un programa, toma la dirección de esa instrucción o dato como su valor. Pero tiene diverso significado cuando está dado la directiva EQU. Entonces toma el operando de EQU como su valor. Las etiquetas se deben poner siempre en la primera columna y se deben seguir por una instrucción (no una línea vacía). Las etiquetas se deben seguir por ': ' (dos puntos), para distinguirlos de la instrucción.

3. Pseudo operaciones (Pseudo Ops)

Solo hay 3 directivas actualmente disponible en el lenguaje ensamblador GNUSIM8085

a. DB - define byte (8 bits)
b. DS - define size (no. of bytes)
c. EQU - like minimalistic #define in C

DB

es usado para definir el espacio para un arreglo de valores especificados separados por coma. Y la etiqueta (Si es dada al inicio de DB) es asignada la dirección del primer dato item. Por ejemplo:

var1: db 34, 56h, 87

"por ejemplo, si se asume que ha incrementado actualmente su registro PC a 4200h, var1=4200h, var1+1=4201h, var1+2=4202h. Observe que "56h" está definido para ser una constante hexadecimal. En este ejemplo se asignan 3 octetos."

DS

se utiliza definir el número especificado de bytes para ser asignado e inicializados a cero. Para tener acceso a cada byte usted puede utilizar el operador ' + ' o ' - ' junto con etiqueta. Por ejemplo,

var2: ds 8

"ahora cuando usted utiliza var2 en el programa, se refiere al primero de estos ocho bytes. Para referirse a otros bytes, como por ejemplo el 3er byte, tiene que utilizar var2+3 en lugar de simplemente var2. Este concepto también ¡se aplica a DB! ' - ' se utiliza para variables previas, es decir, refiere variables anteriores en el programa!"

EQU

se comporta de manera similar a ***#define*** en C. Pero es muy simple. EQU puede ser utilizado solamente para dar nombres a constantes numéricas. No se permite jerarquizar EQU. EQU se puede usar solamente en los operandos para los pseudo operaciones y nemónicos. Por ejemplo,

```
jmp start               ;saltar al código sin pasar por la asignación de datos
;data starts here
port1:  equ   9h
data:   equ   7fh

var1:   db    data, 0   ;como - 7fh, 0

;code starts here
start:  lxi   h, var1   ;carga la dirección de var1 en el par de direccionamiento HL
        mov   a, m      ;carga el contenido de var1 en el registro A (i.e. 7fh in A)
        out   port1     ;envía el contenido del reg A al puerto 9h
        in    port1     ;lee el puerto 9h y lo almanea en el reg A
        sta   var1+1    ;alamacena el contenido del reg A en la locación de memoria var+1 (la
        siguiente de 7fh!)
        hlt             ;termina la ejecución
```

"como podemos ver" EQU define las etiquetas para que puedan ser utilizadas como nombres descriptivos de constantes. Usted debería utilizalas con frecuencia en su programa para evitar números mágicos."

4. Nemónicos[1]

Después de todo, estoy utilizando mi tiempo libre para hacer todas estas cosas. Escribir un manual GRANDE de instrucciones 8085 parece redundante y una pérdida de tiempo. Usted se puede referir muchos libros de texto disponibles para programar en el 8085. :-)

5. Comentarios en la programación

Los comentarios comienzan con un punto y coma (';'). como se puede ver en el ejemplo anterior, los comentarios se pueden dar en cualquier parte de programa. Cualquier cosa después de ';' es sin sentido para el ensamblador, excepto a una secuencia importante de caracteres ...SI LEALO..

Características Extra

1. Puntos de quiebre automáticos

Mientras se familiariza con la aplicación, usted puede utilizar “puntos de quiebre” para depurar errores de su programa. Pero para ciertos programas, se tiene que exhibir algo al usuario antes de continuar. Un ejemplo perfecto para esto es el problema de las N-Queens. Aquí encontrar todas las soluciones para (digamos) 8 reinas consume mucho tiempo (implica un total de 92 soluciones). En mi sistema, tomó casi 1 minuto para encontrar todas las soluciones. Pero en ese proceso solo puedo ver la última solución, puesto que las soluciones se sobreescriben sobre las subsecuentes. Ahora puedo dar un “punto de quiebre” en el lugar donde consigo una solución. Cuando se alcanza el“punto de quiebre”, puedo parar y ver la solución (examinando las variables) y entonces continuar para la solución siguiente.

Pero para este programa, cada vez que lo carga, usted tiene que fijar los “punto de quiebre”. Esto puede ser automatizado. Para fijar los “punto de quiebre” (cuando se carga el programa) en la línea número ' n ', usted tiene que poner un comentario especial en la línea ' n-1 '. Y este comentario debe comenzar en la primera columna. La secuencia es:

```
;@
```

Si se encuentra ";@", el editor fijará el “punto de quiebre” en la línea siguiente. Por razones obvias, usted no puede fijar un “punto de quiebre” en la primera línea en su programa. Por ejemplo, mire el programa N- Queens en la sección de documentación del GNUSIM8085. (nqueens.asm).

1 Comentario del autor del manual Gnusim8085

Notas Finales

No se olvide de incluir la instrucción ***"hlt"*** en alguna parte del programa para terminarlo, si no usted estará ¡ engañado!

Las direcciones constantes se deben utilizar con precaución. ***"lda 2200h"*** será ***"3a 00 22"*** en lenguaje de máquina. Tal que la dirección actual es otra vez 2200h!

Tips

Para Aumentar/Disminuir el tamaño de la fuente en el editor, sostenga presionada la tecla ctrl y gire la rueda del ratón hacia adentro o afuera.

Eso es todo por ahora amigos!
http://gnusim8085.sourceforge.net

ANEXOS

INSTRUCCIONES PARA EL PROCESADOR 8085

Conjunto de instrucciones para el procesador 8085, sintaxis y código de operación agrupadas por función:

TRANSFERENCIA DE DATOS

MOV r1, r2 Copia el contenido del registro r2 al registro r1; r1 <-- r2
No modifica banderas

40h MOV B, B	41h MOV B, C	42h MOV B, D
43h MOV B, E	44h MOV B, H	45h MOV B, L
47h MOV B, A	48h MOV C, B	49h MOV C, C
4Ah MOV C, D	4Bh MOV C, E	4Ch MOV C, H
4Dh MOV C, L	4Fh MOV C, A	50h MOV D, B
51h MOV D, C	52h MOV D, D	53h MOV D, E
54h MOV D, H	55h MOV D, L	57h MOV D, A
58h MOV E, B	59h MOV E, C	5Ah MOV E, D
5Bh MOV E, E	5Ch MOV E, H	5Dh MOV E, L
5Fh MOV E, A	60h MOV H, B	61h MOV H, C
62h MOV H, D	63h MOV H, E	64h MOV H, H
65h MOV H, L	67h MOV H, A	68h MOV L, B
69h MOV L, C	6Ah MOV L, D	6Bh MOV L, E
6Ch MOV L, H	6Dh MOV L, L	6Fh MOV L, A
78h MOV A, B	79h MOV A, C	7Ah MOV A, D
7Bh MOV A, E	7Ch MOV A, H	7Dh MOV A, L
7Fh MOV A, A		

MOV Rd, Rs; Copy from source to destination
MOV M, Rs
MOV Rd, M
This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers.

Example: MOV B, C or MOV B, M

MOV M, Rs Copia el contenido del registro r a la posición de memoria apuntada por el registro HL.
No modifica banderas
[HL] <-- r

70h MOV M, B	71h MOV M, C	72h MOV M, D	77h MOV M, A
73h MOV M, E	74h MOV M, H	75h MOV M, L	

MOV Rd, M Copia al registro r el contenido de la posición de memoria apuntada por el registro HL.

No modifica banderas
r <-- [HL]

46h MOV B, M	4Eh MOV C, M	56h MOV D, M
5Eh MOV E, M	66h MOV H, M	6Eh MOV L, M
7Eh MOV A, M		

MVI Rd, data; Move immediate 8-bit
MVI M, data
The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers. No modifica banderas
Example: MVI B, 57H or MVI M, 57H

R <-- xx

06h MVI B, xx	0Eh MVI C, xx	16h MVI D, xx
1Eh MVI E, xx	26h MVI H, xx	2Eh MVI L, xx
3Eh MVI A, xx		

M <-- xx
36 MVI M, xx

INSTRUCCIONES LXI

01 LXI B
11 LXI D
21 LXI H

INSTRUCCIONES STAX

02 STAX B
12 STAX D

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).
* Código Objeto: 000X 0010 / * Cantidad de Bytes: 1 / * Ciclos de ejecución: 7 / * Flags afectados: --
* Modos de direccionamiento: REGISTRO INDIRECTO

TRANSFERENCIA DE DATOS MEMORIA <--> ACUMULADOR

0A LDAX B
1A LDAX C

32 STA
3A LDA

STAX Reg. pair ; Store accumulator indirect
The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered.
Example: STAX B

LDAX B/D Reg. pair; Load accumulator indirect
The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered.
Example: LDAX B

STA 16-bit address ; Store accumulator direct
The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.
Example: STA 4350H
* Código Objeto: 32 PPQQ / * Cantidad de Bytes: 3 / * Ciclos de ejecución: 13 / * Flags afectados: --
* Modos de direccionamiento: DIRECTO

LDA 16-bit address ; Load accumulator
The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered.
Example: LDA 2034H

22 SHLD
2A LHLD
EB XCHG

SHLD 16-bit address; Store H and L registers direct
The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.
Example: SHLD 2470H
LHLD 16-bit address; Load H and L registers direct
The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of

source memory locations are not altered.

Example: LHLD 2040H

XCHG none; Exchange H and L with D and E

The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.

Example: XCHG

INSTRUCCIONES PARA OPERACIONES DE PILA (STACK)

C5 PUSH B
D5 PUSH D
E5 PUSH H
F5 PUSH PSW

C1 POP B
D1 POP D
E1 POP H
F1 POP PSW

E3 XTHL
F9 SPHL

31 LXI SP
33 INX SP
3B DCX SP

LXI Reg. pair, 16-bit data; Load register pair immediate

The instruction loads 16-bit data in the register pair designated in the operand.

Example: LXI H, 2034H or LXI H, XYZ

SPHL none; Copy H and L registers to the stack pointer

The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered.

Example: SPHL

XTHL none; Exchange H and L with top of stack

The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered.

Example: XTHL

PUSH Reg. pair; Push register pair onto stack

The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high- order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location.

Example: PUSH B or PUSH A

POP Reg. pair; Pop off stack to register pair

The contents of the memory location pointed out by the stack pointer register pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1.

Example: POP H or POP A

INSTRUCCIONES DE SALTO

HEX	Instrucción	Descripción	Estado Bandera
C3	JMP		
DA	JC	Jump on Carry	CY = 1
D2	JNC	Jump on no Carry	CY = 0
CA	JZ	Jump on zero	Z=1
C2	JNZ	Jump on no zero	Z=0
F2	JP	Jump on positive	S=0
FA	JM	Jump on minus	S=1
EA	JPE	Jump on parity even	P=1
E2	JPO	Jump on parity odd	P=0
E9	PCHL		

JMP 16-bit address; Jump unconditionally

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.

Example: JMP 2034H or JMP XYZ

Operand: 16-bit address ; Jump conditionally

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below.

Example: JZ 2034H or JZ XYZ

PCHL none; Load program counter with HL contents

The contents of registers H and L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte.

Example: PCHL

INSTRUCCIONES DE SUBRUTINAS

HEX	Instrucción	Descripción	Estado Bandera
CD	CALL		
DC	CC	Call on Carry	CY = 1
D4	CNC	Call on no Carry	CY = 0
CC	CZ	Call on zero	Z=1
C4	CNZ	Call on no zero	Z=0
F4	CP	Call on positive	S=0
FC	CM	Call on minus	S=1
EC	CPE	Call on parity even	P=1
E4	CPO	Call on parity odd	P=0

CALL 16-bit address; Unconditional subroutine call
The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL the contents of the (program counter) is pushed onto the stack.

Example: CALL 2034H or CALL XYZ

Operand: 16-bit address; Call conditionally
The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below. Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.

Example: CZ 2034H or CZ XYZ

RETORNO DE SUBRUTINAS

C9	RET		
D8	RC	Return on Carry	CY = 1
D0	RNC	Return on no Carry	CY = 0
C8	RZ	Return on zero	Z=1
C0	RNZ	Return on no zero	Z=0
F0	RP	Return on positive	S=0
F8	RM	Return on minus	S=1
E8	RPE	Return on parity even	P=1
E0	RPO	Return on parity odd	P=0

RET none; Return from subroutine unconditionally
The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.

Example: RET

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.

* Código Objeto: C9 / * Cantidad de Bytes: 1 / * Ciclos de ejecución: 10 / * Flags afectados: --
* Modos de direccionamiento: REGISTRO INDIRECTO

Operand: none; Return from subroutine conditionally

The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described below. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.

Example: RZ
Example: RM
Example: RNC, etc...

RZ

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.
* Código Objeto: C8 / * Cantidad de Bytes: 1 / * Ciclos de ejecución: 6/12 / * Flags afectados: --
Modos de direccionamiento: REGISTRO INDIRECTO

RM

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.
* Código Objeto: F8 / * Cantidad de Bytes: 1 / * Ciclos de ejecución: 6/12 / * Flags afectados: --
* Modos de direccionamiento: REGISTRO INDIRECTO

RNC

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.
* Código Objeto: DO / * Cantidad de Bytes: 1 / * Ciclos de ejecución: 6/12 / * Flags afectados: --
- Modos de direccionamiento: REGISTRO INDIRECTO

RNZ

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.
* Código Objeto: CO / * Cantidad de Bytes: 1 / * Ciclos de ejecución: 6/12 / * Flags afectados: --
* Modos de direccionamiento: REGISTRO INDIRECTO

RP

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.
* Código Objeto: FO / * Cantidad de Bytes: 1 / * Ciclos de ejecución: 6/12 / * Flags afectados:

* Modos de direccionamiento: REGISTRO INDIRECTO

RPE
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).
* Código Objeto: B8 / * Cantidad de Bytes: 1 / * Ciclos de ejecución: 6/12 / * Flags afectados: --
* Modos de direccionamiento: REGISTRO INDIRECTO

RPO
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.
* Código Objeto: EO / * Cantidad de Bytes: 1 / * Ciclos de ejecución: 6/12 / * Flags afectados: --
* Modos de direccionamiento: REGISTRO INDIRECTO

RRC
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.
* Código Objeto: OF / *Cantidad de Bytes: 1 / *Ciclos de ejecución: 4 / *Flags afectados: CV
Modos de direccionamiento:

INSTRUCCIONES DE RESTART

C7 RST 0h
CF RST 8h
D7 RST 10h
DF RST 18h
E7 RST 20h
EF RST 28h
F7 RST 30h
FF RST 38h

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.
* Código Objeto: 11XX X111 / * Cantidad de Bytes: 1 / * Ciclos de ejecución: 12
* Flags afectados: -- / * Modos de direccionamiento: REGISTRO INDIRECTO

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:

INSTR.	FORMATO	RST	CONTADOR DE PROGRAMA
RST 0	1100 0111	C7	0000000000000000 0000H
RST 1	1100 1111	CF	0000000000001000 0008H
RST 2	1101 0111	D7	0000000000010000 0010H
RST 3	1101 1111	DF	0000000000011000 0018H
RST 4	1110 0111	E7	0000000000100000 0020H
RST 5	1110 1111	EF	0000000000101000 0028H
RST 6	1111 0111	F7	0000000000110000 0030H
RST 7	1111 1111	FF	0000000000111000 0038H

RST; Restart

0-7 The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. The instructions are generally used in conjunction with interrupts and inserted using external hardware. However these can be used as software instructions in a program to transfer program execution to one of the eight locations.
The addresses are:

The 8085 has four additional interrupts and these interrupts generate RST instructions internally and thus do not require any external hardware. These instructions and their Restart addresses are:

Interrupt	Restart Address
TRAP	0024H
RST 5.5	002CH
RST 6.5	0034H
RST 7.5	003CH

INSTRUCCIONES DE PUERTOS

DB IN n; lee el puerto n
D3 OUT n; escribe en el puerto n

OUT 8-bit port address; Output data from accumulator to a port with 8-bit address

The contents of the accumulator are copied into the I/O port specified by the operand.

Example: OUT F8H

IN 8-bit port address; Input data to accumulator from a port with 8-bit address

The contents of the input port designated in the operand are read and loaded into the accumulator.

Example: IN 8CH

INCREMENTOS Y DECREMENTOS

04 INR B
0C INR C
14 INR D
1C INR E

24 INR H
2C INR L
34 INR M
3C INR A

05 DCR B
0D DCR C
15 DCR D
1D DCR E
25 DCR H
2D DCR L
35 DCR M
3D DCR A

03 INX B
13 INX D
23 INX H

0B DCX B
1B DCX D
2B DCX H

INR R; Increment register or memory by 1
INR M
The contents of the designated register or memory) are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.

E*xample: INR B or INR M*

INX R; Increment register pair by 1
The contents of the designated register pair are incremented by 1 and the result is stored in the same place.

Example: INX H

DCR R; Decrement register or memory by 1
DCR M
The contents of the designated register or memory are decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.

Example: DCR B or DCR M

DCX R; Decrement register pair by 1
The contents of the designated register pair are decremented by 1 and the result is stored in the same place.

Example: DCX H

DAA none; Decimal adjust accumulator

The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.

Example: DAA

INSTRUCCIONES DE SUMA

80 ADD B
81 ADD C
82 ADD D
83 ADD E
84 ADD H
85 ADD L
86 ADD M
87 ADD A

88 ADC B
89 ADC C
8A ADC D
8B ADC E
8C ADC H
8D ADC L
8E ADC M
8F ADC A

C6 ADI
CE ACI
09 DAD B
19 DAD D
29 DAD H

39 DAD SP

ADD R; Add register or memory to accumulator
ADD M

The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.

Example: ADD B or ADD M

ADC R; Add register to accumulator with carry
ADC M

The contents of the operand (register or memory) and the Carry flag are added to the contents of the

accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.

Example: ADC B or ADC M

ADI 8-bit data; Add immediate to accumulator

The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.

Example: ADI 45H

ACI 8-bit data; Add immediate to accumulator with carry

The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.

Example: ACI 45H

DAD Reg. pair; Add register pair to H and L registers

The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. The contents of the source register pair are not altered. If the result is larger than 16 bits, the CY flag is set. No other flags are affected.

Example: DAD H

INSTRUCCIONES DE RESTA

90 SUB B
91 SUB C
92 SUB D
93 SUB E
94 SUB H
95 SUB L
96 SUB M
97 SUB A

98 SBB B
99 SBB C
9A SBB D
9B SBB E
9C SBB H
9D SBB L
9E SBB M
9F SBB A

D6 SUI
DE SBI

SUB M; Subtract register or memory from accumulator
SUB R

The contents of the operand (register or memory) are is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are

modified to reflect the result of the subtraction.
Example: SUB B or SUB M
SUB R; 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.
Código Objeto: 001 0XXX, Cantidad de Bytes: 1, Ciclos de ejecución: 4, Flags afectados: Z, S, P, CY, AC , Modos de direccionamiento: REGISTRO

SUB M; 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.
Código Objeto: 96, Cantidad de Bytes: 1, Ciclos de ejecución: 7, Flags afectados: Z, S, P, CY, AC, Modos de direccionamiento: REGISTRO INDIRECTO

SBB R; Subtract source and borrow from accumulator
SBB M
The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.
Example: SBB B or SBB M

SUI 8-bit data; Subtract immediate from accumulator
The 8-bit data (operand) is subtracted from the contents of theaccumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.
Example: SUI 45H

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.
Código Objeto: D6 YY, Cantidad de Bytes: 2, Ciclos de ejecución: 7, Flags afectados: Z, S, P, CY, AC Modos de direccionamiento: INMEDIATO

SBI 8-bit data; Subtract immediate from accumulator with borrow
The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtracion.
Example: SBI 45H

INSTRUCCIONES LÓGICAS

A0 ANA B
A1 ANA C
A2 ANA D
A3 ANA E
A4 ANA H
A5 ANA L
A6 ANA M
A7 ANA A

A8 XRA B
A9 XRA C
AA XRA D
AB XRA E
AC XRA H
AD XRA L
AE XRA M
AF XRA A

B0 ORA B
B1 ORA C
B2 ORA D
B3 ORA E
B4 ORA H
B5 ORA L
B6 ORA M
B7 ORA A

B8 CMP B
B9 CMP C
BA CMP D
BB CMP E
BC CMP H
BD CMP L
BE CMP M
BF CMP A

E6 ANI
EE XRI
F6 ORI
FE CPI

CMP R; Compare register or memory with accumulator
CMP M

The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved . The result of the comparison is shown by setting the flags of the PSW as follows:

if (A) < (reg/mem): carry flag is set
if (A) = (reg/mem): zero flag is set
if (A) > (reg/mem): carry and zero flags are reset

Example: CMP B or CMP M

CPI 8-bit data; Compare immediate with accumulator

The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:

if (A) < data: carry flag is set
if (A) = data: zero flag is set
if (A) > data: carry and zero flags are reset

Example: CPI 89H

ANA R; Logical AND register or memory with accumulator
ANA M

The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.

Example: ANA B or ANA M

ANI 8-bit data; Logical AND immediate with accumulator

The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.

Example: ANI 86H

XRA R; Exclusive OR register or memory with accumulator
XRA M

The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: XRA B or XRA M

XRI 8-bit data; Exclusive OR immediate with accumulator

The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: XRI 86H

ORA R; Logical OR register or memory with acumulator

ORA M

The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: ORA B or ORA M

ORI 8-bit data; Logical OR immediate with accumulator

The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: ORI 86H

INSTRUCCIONES DE ROTACIÓN

07 RLC
0F RRC
17 RAL
1F RAR

RLC none; Rotate accumulator left

Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D 0 as well as in the Carry flag. CY is modified according to bit D 7. S, Z, P, AC are not affected.

Example: RLC

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
Código Objeto: 07, Cantidad de Bytes: 1, Ciclos de ejecución: 4, Flags afectados: CV, *Modos de direccionamiento:* ---

RRC none; Rotate accumulator right

Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D 7 as well as in the Carry flag. CY is modified according to bit D 0. S, Z, P, AC are not affected.

Example: RRC

RAL none; Rotate accumulator left through carry

Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7. S, Z, P, AC are not affected.

Example: RAL

RAR none; Rotate accumulator right through carry

Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected.

Example: RAR

INSTRUCCIONES ESPECIALES

2F CMA
37 STC
3F CMC
27 DAA

CMA none; Complement accumulator

The contents of the accumulator are complemented. No flags are affected.

Example: CMA

CMC none; Complement carry

The Carry flag is complemented. No other flags are affected.

Example: CMC

STC none; Set Carry

The Carry flag is set to 1. No other flags are affected.

Example: STC

Código Objeto: 37, Cantidad de Bytes: 1, Ciclos de ejecución: 4, Flags afectados: CY, Modos de direccionamiento:

INSTRUCCIONES DE CONTROL

FB EI
F3 DI
00 NOP
76 HLT

NOP none; No operation

No operation is performed. The instruction is fetched and decoded. However no operation is executed.

Example: NOP

HLT none; Halt and enter wait state

The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state.

Example: HLT

DI none; Disable interrupts

The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.

Example: DI

EI none; Enable interrupts

The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to reenable the interrupts (except TRAP).

Example: EI

ESPECIFICAS DEL 8085

20 RIM
30 SIM

RIM none; Read interrupt mask

This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.

Example: RIM

Código Objeto: 29, Cantidad de Bytes: 1, Ciclos de ejecución: 4, Flags afectados: -- , Modos de direccionamiento: ---

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 está a 1
I6.5 = Interrupción 6.5 pendiente si está a 1
I5.5 = Interrupción 5.5 pendiente si está 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 1v

SIM none; Set interrupt mask

This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as follows.

Example: SIM