L'Assembler x86

Istruzioni per la manipolazione dei bit

M. Rebaudengo - M. Sonza Reorda

Politecnico di Torino
Dip. di Automatica e Informatica

Istruzioni per la manipolazione dei bit

Le istruzioni per la manipolazione dei bit si suddividono in:

- ***istruzioni logiche*, che permettono di modificare o controllare uno o più bit**
- ***istruzioni di scorrimento*, che permettono di cambiare la posizione dei bit.**

Istruzioni logiche

Le istruzioni per la manipolazione dei bit permettono di *mascherare* o *forzare* i singoli bit di una parola.

Esempio

Per forzare a `1` il quarto bit di `AX` senza modificare gli altri bit:

```
OR          AX, 1000b
```

Per saltare quando il settimo bit di `TAB` vale `0`:

```
TEST        TAB, 1000000b
JZ          TUTTO_ZERO
```

Istruzione AND

Formato

```
AND  dest, sorg
```

Funzionamento

L'istruzione *AND* esegue l'operazione logica AND bit a bit tra il contenuto dell'operando *dest* ed il contenuto dell'operando *sorg*. Il risultato dell'operazione è copiato in *dest*.

L'operando *dest* può essere un registro od una locazione di memoria.

L'operando *sorg* può essere un registro, una locazione di memoria oppure un valore immediato.

L'operando *sorg* è detto *maschera*.

Conversione da codice ASCII a numero binario

```
                .MODEL      small
                .STACK
                .DATA
ASCIIDB         ?
NUM             DB      ?
                ...
                .CODE
                ...
                MOV   AL, ASCII
                AND   AL, 0FH     ; mascheramento dei
                                  ; 4 bit alti
                MOV   NUM, AL
                ...
```

binario	oct	dec	hex	
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;

L'istruzione OR

Formato

```
OR   dest, sorg
```

Funzionamento

L'istruzione *OR* esegue l'operazione logica OR bit a bit tra il contenuto dell'operando *dest* ed il contenuto dell'operando *sorg*.

Il risultato dell'operazione è copiato in *dest*.

L'operando *dest* può essere un registro od una locazione di memoria.

L'operando *sorg* può essere un registro, una locazione di memoria oppure un valore immediato.

L'operando *sorg* è detto *maschera*.

Conversione da numero binario a codice ASCII

```
                .MODEL     small
                .STACK
                .DATA
ASCIIDB         ?
NUM             DB     ?
                ...
                .CODE
                ...
                MOV    AL, NUM
                OR     AL, 30H      ; mascheramento dei
                                    ; 4 bit alti
                MOV    ASCII, AL
                ...
```

binario	oct	dec	hex		
010 1110	056	46	2E	.	10
010 1111	057	47	2F	/	10
011 0000	060	48	30	0	10
011 0001	061	49	31	1	10
011 0010	062	50	32	2	10
011 0011	063	51	33	3	10
011 0100	064	52	34	4	10
011 0101	065	53	35	5	10
011 0110	066	54	36	6	10
011 0111	067	55	37	7	10
011 1000	070	56	38	8	10
011 1001	071	57	39	9	10
011 1010	072	58	3A	:	10
011 1011	073	59	3B	;	10

L'istruzione XOR

Formato

```
XOR  dest, sorg
```

Funzionamento

L'istruzione *XOR* esegue l'operazione logica EXOR bit a bit tra il contenuto dell'operando *dest* ed il contenuto dell'operando *sorg*. Il risultato dell'operazione è copiato in *dest*.

L'operando *dest* può essere un registro od una locazione di memoria.

L'operando *sorg* può essere un registro, una locazione di memoria oppure un valore immediato.

L'operando *sorg* è detto *maschera*.

L'istruzione NOT

Formato

NOT operando

Funzionamento

L'istruzione *NOT* esegue l'operazione logica di complementazione bit a bit del contenuto dell'*operando*.

L'*operando* può essere un registro od una locazione di memoria.

L'istruzione `TEST`

Formato:

```
TEST        dest, sorg
```

Funzionamento:

L'istruzione `TEST` esegue l'operazione logica AND bit a bit tra il contenuto dell'operando *`dest`* ed il contenuto dell'operando *`sorg`*, senza modificare il contenuto degli operandi. L'istruzione aggiorna coerentemente il flag `ZF`.

L'operando *`dest`* può essere un registro od una locazione di memoria.

L'operando *`sorg`* può essere un registro, una locazione di memoria oppure un valore immediato.

L'operando *`sorg`* è detto *maschera*.

Testa o Croce ?

```
        .MODEL  small
        .STACK
        .DATA
TESTA   DB "TESTA",0Dh,0Ah,"$"
CROCE   DB "CROCE",0DH,0AH,"$"
        .CODE
        ...
        MOV  AH, 2CH
        INT  21H       ; in DX il timer di sistema
        TEST DH, 1     ; bit 0 = 0 ?
        JNZ  lab_t     ; No: va a lab_t
        LEA  DX, CROCE    ; Sì: in DX l'offset di CODA
        JMP  video
lab_t:  LEA  DX, TESTA    ; in DX l'offset di TESTA
video:  MOV  AH, 09H
        INT  21H       ; visualizza su video
```

Istruzioni di scorrimento

Le *istruzioni di scorrimento* permettono di modificare la posizione dei bit all'interno di una parola, spostandoli verso sinistra o verso destra di un numero definito di posizioni.

Le istruzioni di scorrimento si dividono in due gruppi:

- ***istruzioni di shift*, in cui l'ultimo bit nella direzione dello scorrimento è posto a 0 o ad un valore uguale a quello del bit di segno;**
- ***istruzioni di rotazione*, in cui l'ultimo bit nella direzione della rotazione viene copiato al posto del primo bit.**

Istruzioni di scorrimento

(segue)

Il formato delle istruzioni di scorrimento è il seguente:

`OPCODE operando, contatore`

`operando` **può essere**

- **un registro oppure**
- **una locazione di memoria**

`contatore` **può essere**

- **il valore immediato `1` oppure**
- **il registro `CL`.**

A partire dall'80186 è possibile utilizzare come *`contatore`* qualunque valore immediato.

Le istruzioni SHL e SHR

Formato:

SHL operando, contatore

SHR operando, contatore

Funzionamento:

L'istruzione *SHL* esegue lo scorrimento a sinistra del contenuto dell'*operando* di un numero di posizioni pari al valore di *contatore*.

L'istruzione *SHR* esegue lo scorrimento a destra del contenuto dell'*operando* di un numero di posizioni pari al valore di *contatore*.

L'ultimo bit in uscita viene copiato nel flag `CF`; tutte le posizioni vuote vengono caricate con bit di valore `0`.


CF
operando
0
SHL
0
operando
CF
SHR

Le istruzioni SHR e SHL

(segue)

L'istruzione *SHR* equivale ad una divisione per 2^n nel caso di numeri interi senza segno.

L'istruzione *SHL* equivale ad una moltiplicazione per 2^n nel caso di numeri interi senza segno.

Le istruzioni SAL e SAR

Formato

SAL operando, contatore

SAR operando, contatore

Funzionamento

L'istruzione SAL esegue lo scorrimento a sinistra del contenuto dell'*operando* di un numero di posizioni pari al valore di *contatore*. I bit vuoti a destra sono riempiti di bit a 0.

L'istruzione SAR esegue lo scorrimento a destra del contenuto dell'*operando* di un numero di posizioni pari al valore di *contatore*. I bit vuoti a sinistra sono riempiti di bit pari al valore del bit più significativo.

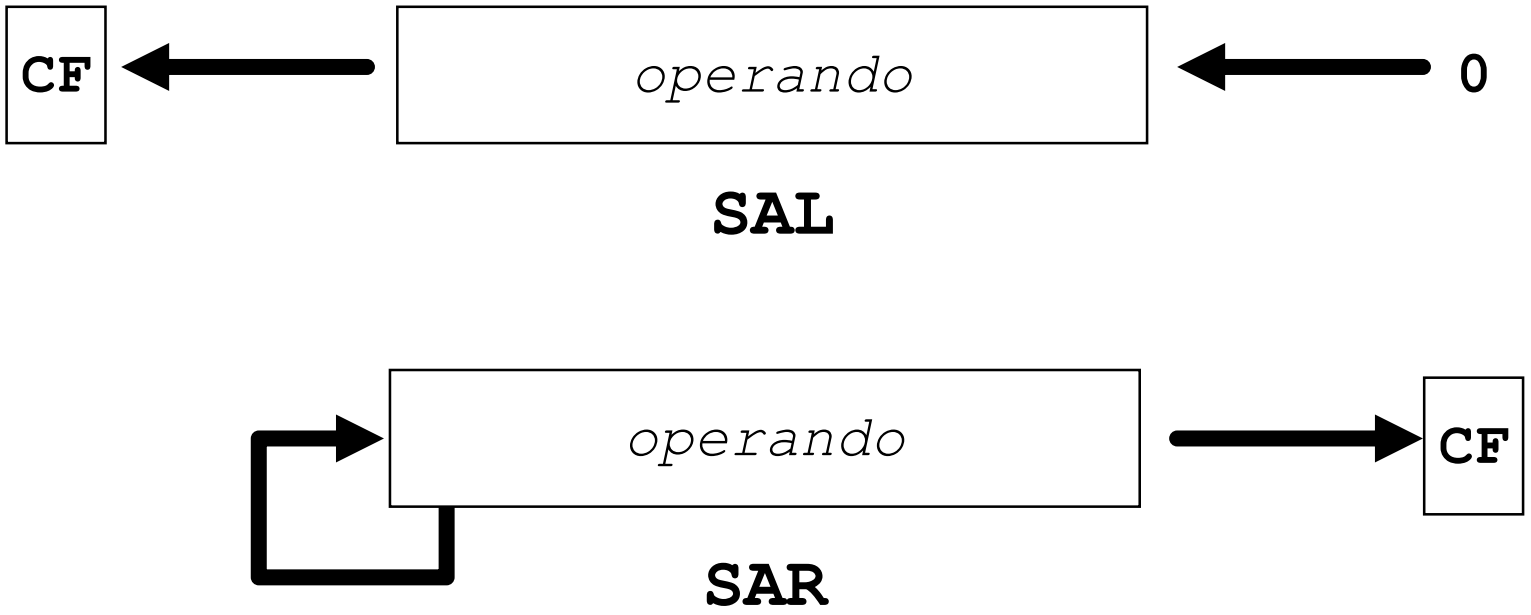
Le istruzioni SAL e SAR

(segue)

L'istruzione `SAL` è del tutto equivalente all'istruzione `SHL`.

L'istruzione `SAR` permette di eseguire l'operazione di divisione di un numero intero con segno per una potenza di 2.

Per entrambe le istruzioni, l'ultimo bit in uscita viene copiato nel flag `CF`.


CF
operando
0
SAL
operando
CF
SAR

Le istruzioni ROR e ROL

Formato:

ROL operando, contatore

ROR operando, contatore

Funzionamento:

Le istruzioni ROR e ROL permettono di eseguire la rotazione del contenuto di un operando.

L'ultimo bit in uscita viene copiato nel flag CF.

In una rotazione a destra (ROR) in CF è copiato il bit meno significativo, mentre in una rotazione a sinistra (ROL) in CF è copiato il bit più significativo.

Scambio del contenuto dei nibble in un byte

```
        .MODEL      small
        .STACK
        .DATA
NUM1    DB      ?
NUM2    DB      ?
        .CODE
        ...
        MOV     AL, NUM1
        MOV     CL, 4
        ROL     AL, CL          ; rotazione a sinistra di 4 bit
        MOV     NUM2, AL
        ...
```

Le istruzioni RCR e RCL

Formato

RCL operando, contatore

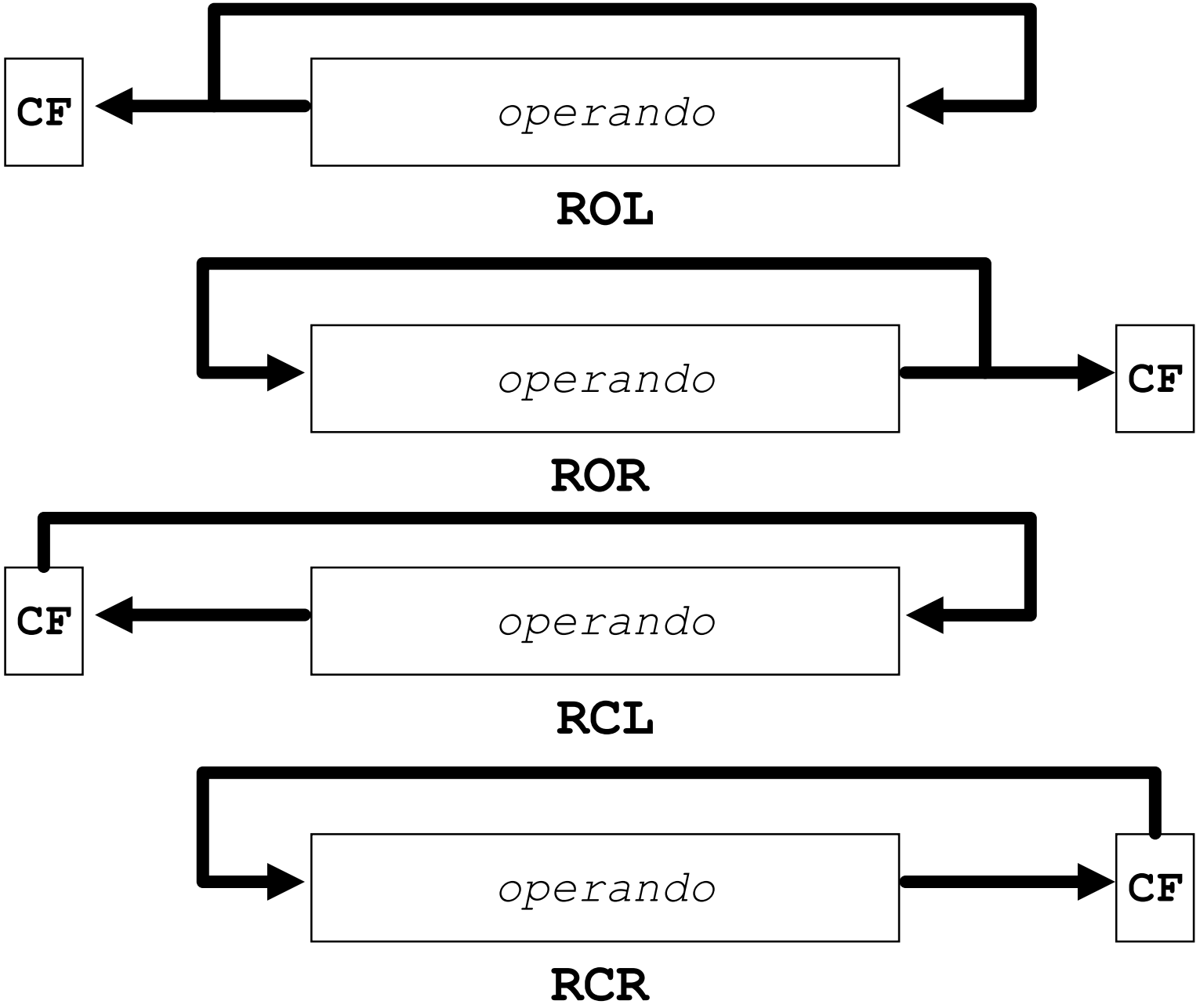
RCR operando, contatore

Funzionamento

Le istruzioni RCR e RCL permettono di eseguire la rotazione del contenuto di un *operando* utilizzando il flag CF come bit aggiuntivo.

In una rotazione a destra (RCR) la rotazione avviene come se CF fosse un bit in più posto alla destra della parola da ruotare.

In una rotazione a sinistra (RCL) la rotazione avviene come se CF fosse un bit in più posto alla sinistra della parola da ruotare.


CF
operando
ROL
operando
CF
ROR
CF
operando
RCL
operando
CF
RCR

Calcolo dell'area di un triangolo

```
        .MODEL      small
        .STACK
        .DATA
BASE    DW    ?
ALT     DW    ?
AREA    DW    ?
        .CODE
        ...
        MOV   AX, BASE
        MUL   ALT    ; DX,AX = BASE * ALTEZZA
        SHR   DX, 1 ; carico CF con il bit 0 di DX
        RCR   AX, 1 ; divisione per 2 e copia di CF nel bit 15
        CMP   DX, 0 ; DX != 0 ?
        JNE   err    ; Sì: overflow
        MOV   AREA, AX
        ...
err:    ...          ; gestione dell'overflow
```