

# L'Assembler 8086

## Istruzioni per la Manipolazione dei Bit

M. Rebaudengo - M. Senza Reorda

Politecnico di Torino  
Dip. di Automatica e Informatica

1

M. Rebaudengo, M. Senza Reorda

## 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.

2

M. Rebaudengo, M. Senza Reorda

## 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
```

3

M. Rebaudengo, M. Senza Reorda

## 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*.

4

M. Rebaudengo, M. Senza Reorda

## Conversione da codice ASCII a numero binario

```

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

```

5

M. Rebaudengo, M. Sonza Reorda

## 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*.

6

M. Rebaudengo, M. Sonza Reorda

## Conversione da numero binario a codice ASCII

```

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

```

7

M. Rebaudengo, M. Sonza Reorda

## 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*.

8

M. Rebaudengo, M. Sonza Reorda

## 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.

9

M. Rebaudengo, M. Sonza Reorda

## 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*.

10

M. Rebaudengo, M. Sonza Reorda

## 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

```

11

M. Rebaudengo, M. Sonza Reorda

## 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.

12

M. Rebaudengo, M. Sonza Reorda

## Istruzioni di scorrimento

(segue)

Il formato delle istruzioni di scorrimento è il seguente:

*OPCODE operando, contatore*

L'*operando* può essere o un registro oppure una locazione di memoria; il *contatore* può essere o il valore immediato 1 oppure il registro CL.

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

13

M. Rebaudengo, M. Sonza Reorda

## 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.

14

M. Rebaudengo, M. Sonza Reorda



15

M. Rebaudengo, M. Sonza Reorda

## Le istruzioni SHR e SHL

(segue)

L'istruzione *SHR* equivale ad una divisione per  $2^n$ .

L'istruzione *SHL* equivale ad una moltiplicazione per  $2^n$ .

16

M. Rebaudengo, M. Sonza Reorda



## 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.

17

M. Rebaudengo, M. Sonza Reorda

## Le istruzioni SAL e SAR

(segue)

L'istruzione *SAL* permette di eseguire l'operazione di moltiplicazione di un numero intero con segno per una potenza di 2. 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 eseguire correttamente l'operazione di divisione occorre eseguire l'estensione del segno verso il bit più significativo.

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

18

M. Rebaudengo, M. Sonza Reorda



19

M. Rebaudengo, M. Sonza Reorda

## 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.

20

M. Rebaudengo, M. Sonza Reorda

## 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
...
```

21

M. Rebaudengo, M. Sonza Reorda

## Le istruzioni RCR e RCL

### Formato

*RCL operando, contatore*

*RCR operando, contatore*

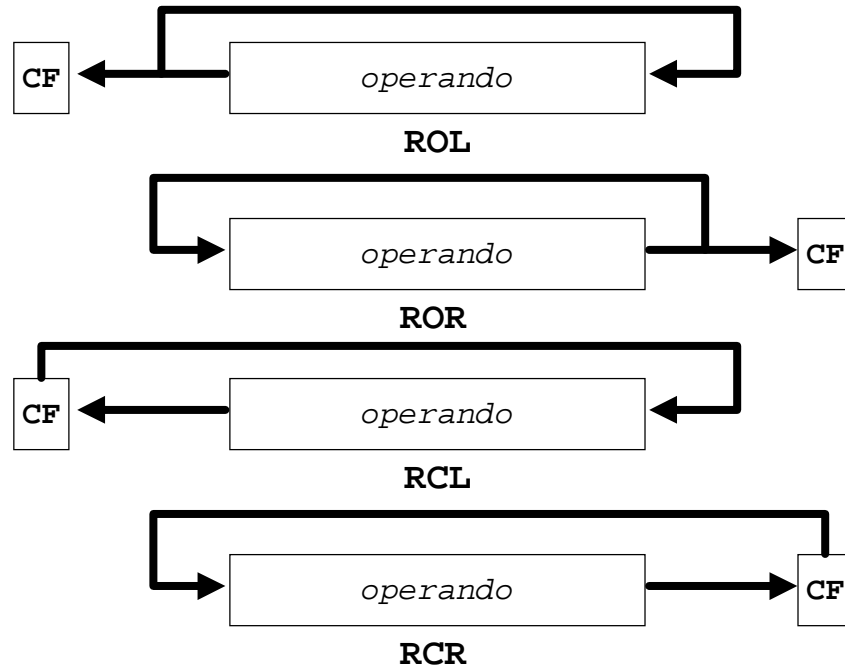
### Funzionamento

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

22

M. Rebaudengo, M. Sonza Reorda



23

M. Rebaudengo, M. Sonza Reorda

## 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: ...   ; istruzioni di gestione dell'overflow

```

24

M. Rebaudengo, M. Sonza Reorda